Faculty of Engineering and Science

Industrial IT

3-Phase Separator

Mathias Lustenberger, Emanuel Matt,
Catalin Marian Orzan and Robin Jesse Schwarz

supervised by
Muhammad Faisal Aftab

November 17, 2023

Contents

List of Figures

List of Tables

1 Introduction

This project is dedicated to programming a Beckhoff PLC for an oil separator system *(Figure 1.1)*. The system represents a three-phase separator, already equipped with a Beckhoff PLC, connected to several components, including three pressure transmitters (PT1-3), two Pressure Differential Transmitters (PTD1, PTD2), a Temperature Transmitter (TT1), two Level Transmitters (LT1, LT2), one Flow Valve (FV1), and two Level Valves (LV2 and LV3). The three main components of the system in question are: the First Stage Separator, the Second Stage Separator and the Gas Scrubber.

Using TwinCAT software, our aim is to create a control program for this system to ensure the optimal performance and minimize the operational downtime. The program will manage the pre-mixed water, oil, and gas supplied to the equipment, guiding them through a gravity separation unit. Ultimately, the PID controller will ensure the efficient separation of oil, water, and gas. The PLC continuously monitors sensor data for the water level, pressure and temperature and the PID controller keeps a constant level so that the water will stay on top, the oil being separated to the right side.

This project provides a valuable opportunity to dive into the industrial side of programming, offering a real-life scenario that incorporates both mechanical knowledge and software aspects. It also expands our understanding of programming in structured environments.

Our team, that consists of four exchange students – Mathias Lustenberger, Emanuel Matt, Catalin Marian Orzan, and Robin Jesse Schwarz, is utilizing knowledge gained from MAS247 lectures and personal research to accomplish project objectives. Each member is responsible for various tasks, including program development, creating multiple HMI (Human Machine Interface) versions for the user, programmer and supervisor, implementing a PID controller and fine-tuning the values, optimizing the system, and compiling a suitable report.

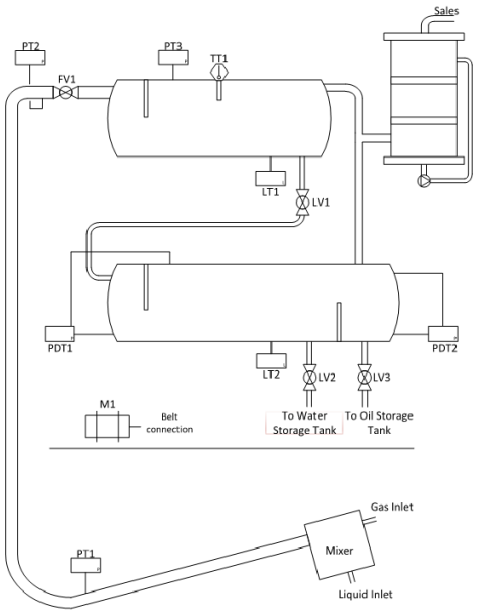

Figure 1.1: Schematics of the Oil Separator System

2 Methods

The project was approached with a systematic methodology in a collaborative manner. We began by reviewing the course materials and lab exercises relevant for our project, and attendance to the project classes provided us with valuable guidance. Then following steps were established: initial research and flowchart prototype, gathering information and documentation, task distribution, development of the code and the PID parameters, development of the HMI design, report and the FAT IAT paperwork, concluded by iterative development and refinement.

2.1 Research and Flowchart Prototype

The project began with a thorough research for all the materials needed. We began by collecting useful documentation, including technical documentation for the Beckhoff CX5240 PLC *(Figure 2.1)*[1], 266HSH Gauge and Pressure Transmitter *(Figure 2.2)*[2], Magnetostrictive Level Transmitter *(Figure 2.3)*[3], 2600T Pressure Transmitter *(Figure 2.4)*[4], AT500 SensyTemp Resistance Thermometer *(Figure 2.5)*[5], TF4100 TC3 Controller Toolbox for the PID[6], and oil data (Mobil DTE 24 ISO VG 32). Additionally, we gathered multiple useful software documentation from Beckhoff Infosys.

Figure 2.1: Beckhoff CX5240 PLC

Figure 2.2: 266HSH Gauge and Pressure Transmitter

Figure 2.3: AT500 Magnetostrictive Level Transmitter

Figure 2.4: 2600T Pressure Transmitter

Figure 2.5: SensyTemp Resistance Thermometer

Alongside the provided TwinCAT software for programming the PLC, we utilized the Overleaf cloud-based LaTeX editor for real-time collaboration on report writing. We utilized diagrams.net, a cross-platform graph drawing software, for creating flowcharts. MATLAB software was used to simulate the system and tune the PID parameters, in Simulink. Additionally, we utilized communication tools such as WhatsApp to stay connected within a group dedicated to the project, Microsoft OneDrive file hosting in order to share project files and a shared Excel Spreadsheet for meetings attendance to ensure a proper organization and Adobe Photoshop to cut and edit photos for the HMI and report. A Gantt chart, shown in the last *(Figure 6.3)* of the report's Appendix.

To ensure an accurate implementation and compatibility of our project, we continued by putting the basic information together in a Flowchart *(Figure 2.6)*, in order to be approved by the teacher. The initial workflow considered the main program, clarifying how it should work. This was followed by a structured approach to implementing safety features before starting any actual work on the project. The safety features include factors such as temperature and pressure. When these values exceed specific thresholds, an alarm will be triggered, and the system will behave accordingly. Additionally, there is a physical safety button that can halt the system at any given time.

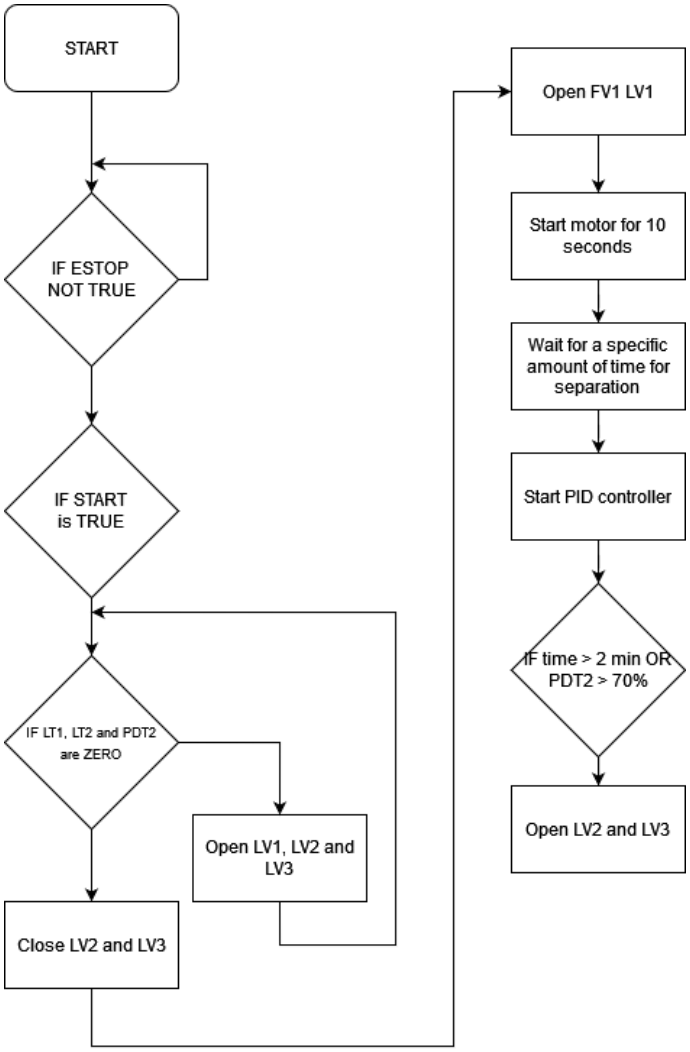

Figure 2.6: Flowchart

2.2 Code and PID Development

With the provided skeleton project in hand, we started the program development phase. We opened the file in TwinCAT and connected the physical PLC to a laptop running the program. First, we set the Real-Time values to 4 cores [7], matching the PLC capabilities. Then, we selected the target system for our PLC and configured them to share the same IP address by clicking Search(Ethernet)... button and selecting the required one from there. After establishing a connection, with the required safety gear equipped, we tested the system by turning on valves, activating the pump's motor, and reading values from sensors to understand which variables they were linked to. A MAIN program was already existing in the skeleton with the variables for the valves, motor control and sensors, together with a demo Human Machine Interface.

With the remark of having PDT2 (Differential Pressure Transmitter) and PT3 (Pressure Transmitter) physically faulty and the LV2 (Level Valve) partially clogged, we adapted and started writing our code accordingly. Another problem encountered was the foam - the process of mixing the oil generates a foam, which causes the system to not run at its maximum potential.[8]

The targeted system was an efficient oil separator, which function will be achieved by the second thank. The first tank serves the purpose of a buffer, adding water to the left side of the second tank, to maintain the oil on top, slowly flowing to the right side of the tank. Our program must be started from a start button. A check if all tanks are empty occurs, based on LT1 and LT2. In case of liquid found in any of the tanks, a draining process takes place. If the tanks are empty, LV1 valve opens and the LV2 and LV3 valves close, and the motor pump starts for a specific amount of time. After this step, the water-oil mixture sits still, ready for gravity to naturally separate them due to their different density. At this point, the PID controller takes charge. The valve (LV1) of the buffer tank partially closes to control the filling speed, while LV2 valve opens, initiating the liquid pumping by the motor pump. The PID controller actively adjusts LV1 and LV2 valves, managing the flow according to the level. A subprogram always maintains the level of the buffer tank slightly filled, based on LT1 sensor data. As we near the end of the process, LV2 and LV3 valves open, efficiently achieving the separation of the oil, accurately checking the empty level with data provided by PDT1.[9]

The code is set up with a Program Organization Unit (POU) in the form of a Function Block Diagram (FBD). Each step has its own action block. We used transitions to check if specific conditions were met, like pressing a button or if a sensor value is above or below a certain point. Also, warning steps are checked after every step. The variables are organized into six Data Unit Types (DUTs) for the PID controller, valves control, sensor data, HMI, motor control, and user interactions with buttons. We require a set of global variables to retain essential information concerning the user, motor, valve, PID, HMI, and warnings.
Safety was a top priority in our project. We programmed the emergency button to work right from the start, stopping the motor and closing all valves. The stop button can pause the system whenever needed. Safety sensors trigger an alarm if PT1 or PT2 goes above $4bar$ or if TT exceeds $80°C$. The temperature transmitter is also set to turn off the motor and open all valves to release pressure. If the pressure gets too high, the motor stops, and flow valve 1 opens.

The design of the PID-Controller is based on equation (1), which determine the change in height of the liquid based on the in and outflow of the tank.

$$\Delta h = \frac{1}{A} * \int (Q_{in} - Q_{out})\ \mathrm{d}t \tag{1}$$

$$Q_{max} = \frac{1}{2} r^2 \pi * \sqrt{2gh_m} \tag{2}$$

With equation (1) and the limits of the valves a system was designed, simulating the controlling of the mixtures level in the tank, shown in figure 2.7.

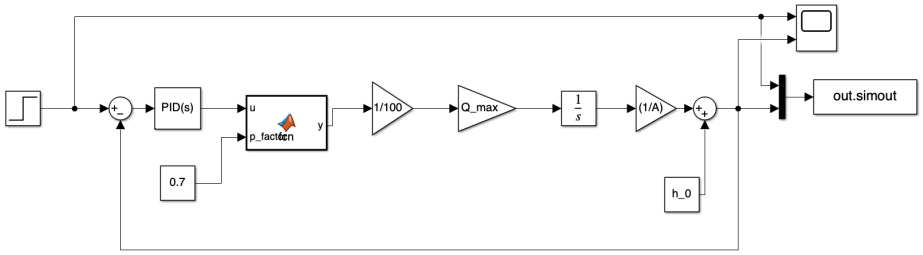

Figure 2.7: Simulink Model of PID-Controller

By giving the system the height aimed for as input, the system returns the current height, which is then subtracted from the aimed height giving the system the current difference in height. The Parameters for the system where either determined by measuring on the physical machine or assumptions, shown in table 1.

Name	**Value**	**Source**
Average area of tank in controlled height $[A]$	270'000 mm	Figure 6.2
Height after filling 2nd tank $[h_0]$	155 mm	assumption
Radius of valve $[r]$	12.5 mm	measured
Gravitational acceleration $[g]$	9.806 m/s^2	
Height of liquid above valve (norm. to 1m) $[h_m]$	1 m	assumption

Table 1: System Variables

The function block is defined so the output valves is controlled depending on the height difference in the tank. The input valves is set to 70% and only changes its value when the height difference in the tank is negative (tank is filled more than aimed for)(*Figure 6.1*). The factor 6 for output and 18 for input were defined by testing on the physical system.

2.3 HMI and FAT IAT Document

We developed distinct versions of Human Machine Interface (HMI) to accommodate diverse operational scenarios. The primary screen provides an overarching view of the system, featuring a secure login option. Upon password entry, users can access one of three HMI versions.

The first user version features buttons for emergency stop, normal stop (pause step), and tank emptying, along with a logout option. An illustration of the system offers a real-time preview. A trend button leads to a detailed view with graphical representations of level transmitters, temperature, pressure differential transmitters, and pressure transmitter values.
The second version accessible through the login screen, serves as the programming interface. It empowers users to modify parameters, including PID parameters, task and control times, limits for integration and output, etc. The interface includes buttons for setting values, resetting, and help.
The third version, designed for supervisory functions, similar the programming interface, but is oriented towards controlling system components. It enables the manipulation of valve opening levels, motor speed, and state, while also providing visualizations of data from all sensors.

In emergency scenarios, a dedicated HMI automatically pops up when pressure or temperature exceeds safety thresholds, or when the emergency button is pressed. This interface displays the triggering values, together with the expected normal values, and a reset option.

3 Result and Discussion

3.1 Program development

The coding process started with a series of changes to the MAIN function, provided in the Skeleton file. The variables provided were written in global sets and a series of conversions (Figure 3.2) took place, together with calling the Run and Blinker functions. The resulted program has the main POU called "Run", under the form of a Functional Block Diagram, that is represented in Figure 3.1.

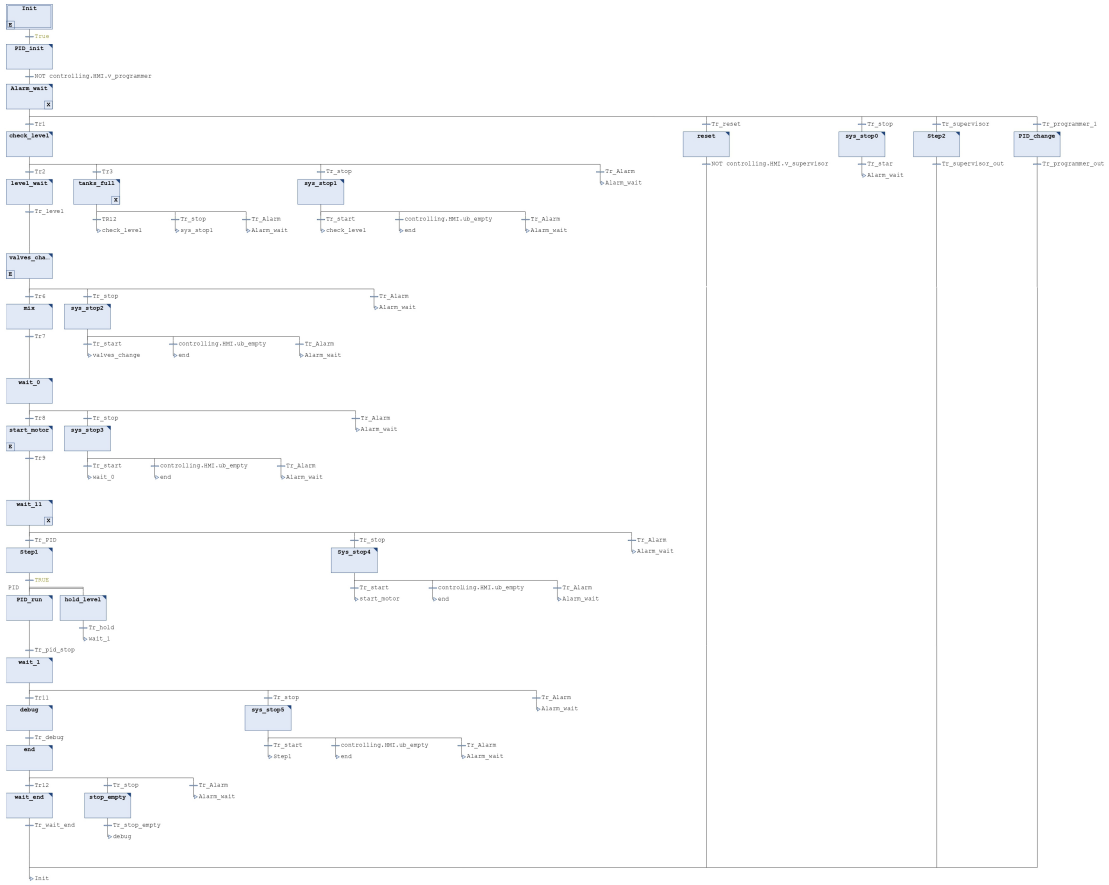

Figure 3.1: The Run Structure

```
//calculate and write them to global
values.data.level_lt1       := Level_Transmitter_1*304.8/32767;                    //measures the fluidlevel in tank 1 in mm
values.data.level_lt2       := Level_Transmitter_2*304.8/32767;                    //measures the fluidlevel in tank 2 in mm
values.data.temp_tt         := (Temperature_Transmitter*300/32767)-50+5;           //measure temperature tank 1 [-50, 250] celsius workspace 16bit + Offset
values.data.pressure_pdt1   := (Pressure_Differencial_Transmitter_1*396/32767+4)/2;  //in mbar
values.data.pressure_pdt2   := (Pressure_Differencial_Transmitter_2*396/32767+4)/2;  //in mbar
values.data.pressure_pt1    := (10*Pressure_Transmitter_1*(600-6))/32767;          //measure pressure bottom pipe [6kps, 600kps] workspace 16 bit //calculate in mbar
values.data.pressure_pt2    := (10*Pressure_Transmitter_2*(600-6))/32767;          //measure pressure bottom pipe [6kps, 600kps] workspace 16 bit //calculate in mbar;
values.data.pressure_pt3    := (10*Pressure_Transmitter_3*(600-6))/32767;          //measure pressure bottom pipe [6kps, 600kps] workspace 16 bit //calculate in mbar;
values.data.level_oil       := values.data.pressure_pdt2*100/(9.81*800);
//callculate the height depending on pdt2
level_oil                   := values.data.pressure_pdt2*100/(9.81*8);
level_mix                   := values.data.pressure_pdt1*100/(9.81*5);
```

Figure 3.2: Initial Conversions

Every variable a crucial role in managing different aspects of the system, ensuring clarity and ease of use. This structured approach not only simplifies the current handling of the system but also facilitates future code modifications and improvements. Insight of the program's structured variables and their descriptions are shown through Table 2, providing an overview of their intercon-

nections. Additionally, global variables are needed to store various parts of the TwinCAT program and make them accesible throughout the entire software, as detailed in Table 3. This systematic organization not only enhances current usability but also sets the groundwork for future enhancements, ensuring the program's continual improvement.

Name	**Variables**	**Description**
st_control	it_flow_valve_1, it_level_valve_1, it_level_valve_2, it_level_valve_3, it_gas_valve, it_mixer_valve	Structure for the valves variable
st_motor	it_flow_valve_1, it_level_valve_1, it_level_valve_2, it_level_valve_3, it_gas_valve, it_mixer_valve	Structure for the variables connected to the motor
st_user	bstart_button, bstop_button, bemergency, bIsInitalized	Structure for the the buttons users use to interact with the system.
st_data	it_flow_valve_1, it_level_valve_1, it_level_valve_2, it_level_valve_3, it_gas_valve, it_mixer_valve	Structure for the transmitters data

Table 2: Structures of variables

Name	**Variable(S)**	**Description**
controlling	user, motor, valve, pid_stparams, call_pid, HMI	Variables used to store structures for control
values	data	Variable used to store data structure
warn	error_press, error_temp, error_ebut	Variables to store alarm data.

Table 3: Global variables

Additionally, two structures were required to store the PID data and the HMI information. Transitions were used to check if specific con-ditions were met, like pressing a button or if a sensor value is above or below a certain point. Also, warning steps are checked after every step in the transitions. Function blocks with active, entry and exit functions allow the program to execute specific functions, such as opening/closing the valves, interpreting sensors data and updating the status of the system.[10]

3.2 PID-Controller

Based on the initial design of the system described in Section 2.2, the PID-Values Kd, Tv, Td and Tn were defined by testing on the physical system (shown in Table 4). Due to the output valve's issue with clogging, an additional performance parameter was added (shown $p\_{factor} = 0.7$ in figure 2.7) [11]
[12].

Name	Value
Kp	0.1
Tv	0
Tn	0
Td	10 ms

Table 4: PID-Controller

With the PID-Controller implemented, the Step response of the system was calculated using a change in the aimed height from 155 mm to 160 mm.

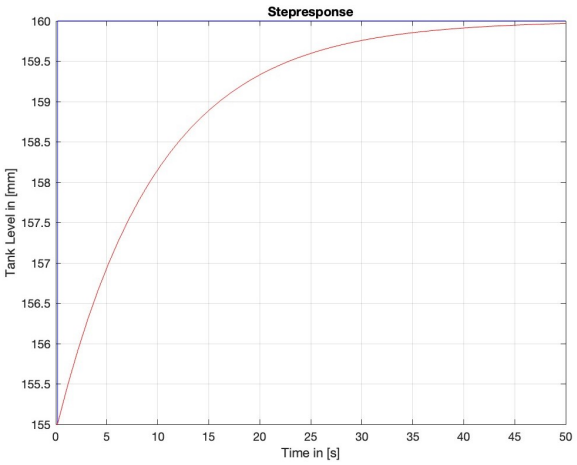

Figure 3.3: Step response of target height change from 155 mm to 160 mm

3.3 Human Machine Interfaces

HMIs play a crucial role in PLC software. They are the user-friendly face of the machine. It provides a visual display on a screen, showing buttons and data that allow operators to control and monitor the industrial processes.

Without a well-designed HMI, operators would struggle to interact with the PLC effectively, therefore they are used to navigating a complex system without a user manual. HMI simplifies the interaction, ensuring seamless communication between humans and machines. In essence, it serves as the control panel, making the operation and supervision of industrial processes more efficient and accessible.

The HMI main screen consist of a Login menu 3.4. Once the Login button is pressed, a numerical keyboard will popup. You can use the following pincodes to access the different HMI versions: 1111 for user, 3333 for supervisor and 6666 for programmer.

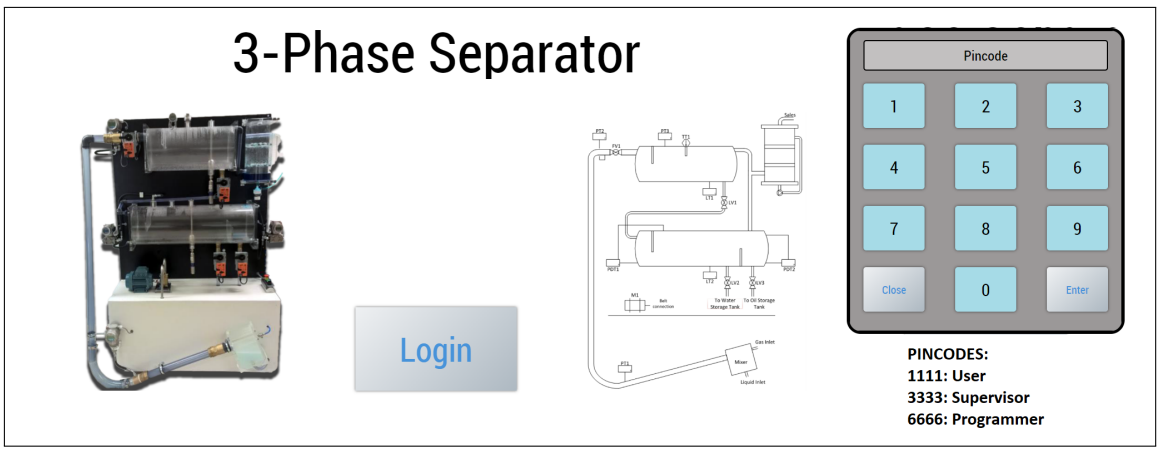

Figure 3.4: Login Human Machine Interface

The first HMI is the User variant (Figure 3.5), accessed by '1111' pincode. It's used to show a visual representation of what's happening with the actual system, with real time adjustments. The interactions with it are very easy, buttons being self-explanatory and proeminent[13][14].

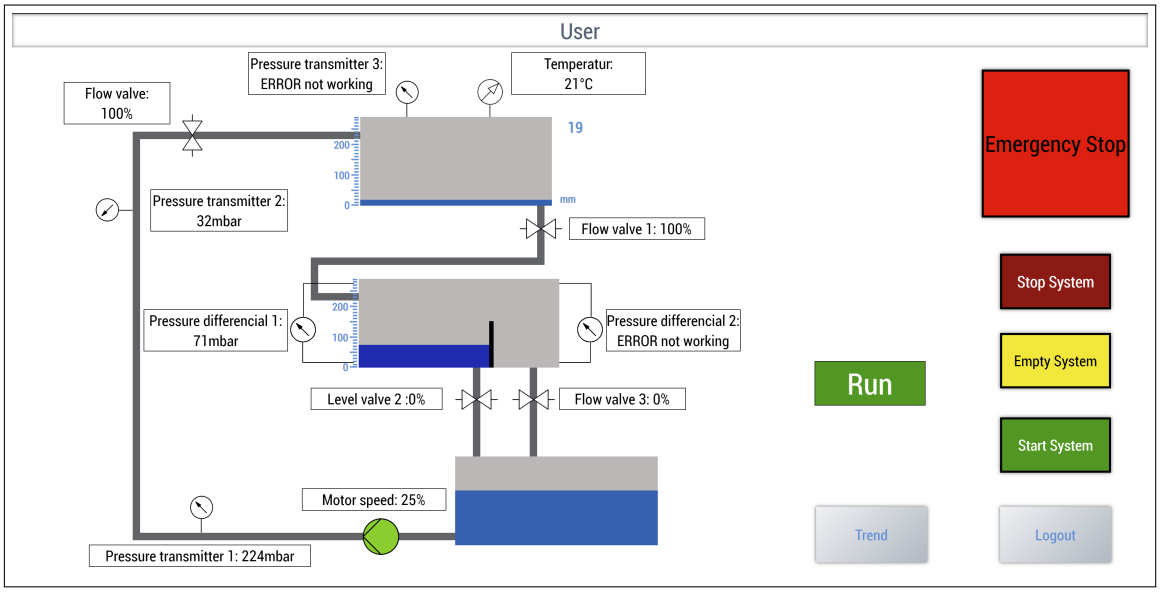

Figure 3.5: User's Human Machine Interface

A useful menu that can be accessed through the User HMI is the Trends HMI (Figure 3.6). This consists of four graphs, showing real time plotted data about multiple transmitters.

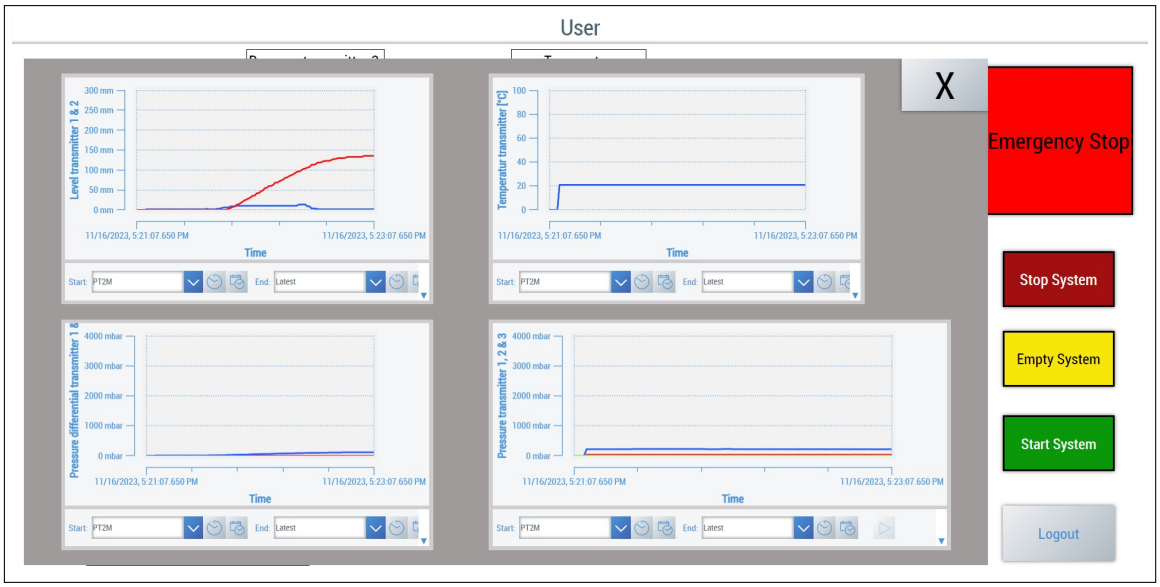

Figure 3.6: Trends Human Machine Interface

Pincode 6666 leads to the programmer's HMI (Figure 3.7). Here, PID values can be ajusted [15], together with other parameters. The "Evil" button is the result of a bug, which should have been a redirect towards a help page on the beckhoff website in order to set parameters, but it ended up being a webpage that can not be closed in Beckhoff TwinCAT HMI system, tehrefore shoudld not be pressed.

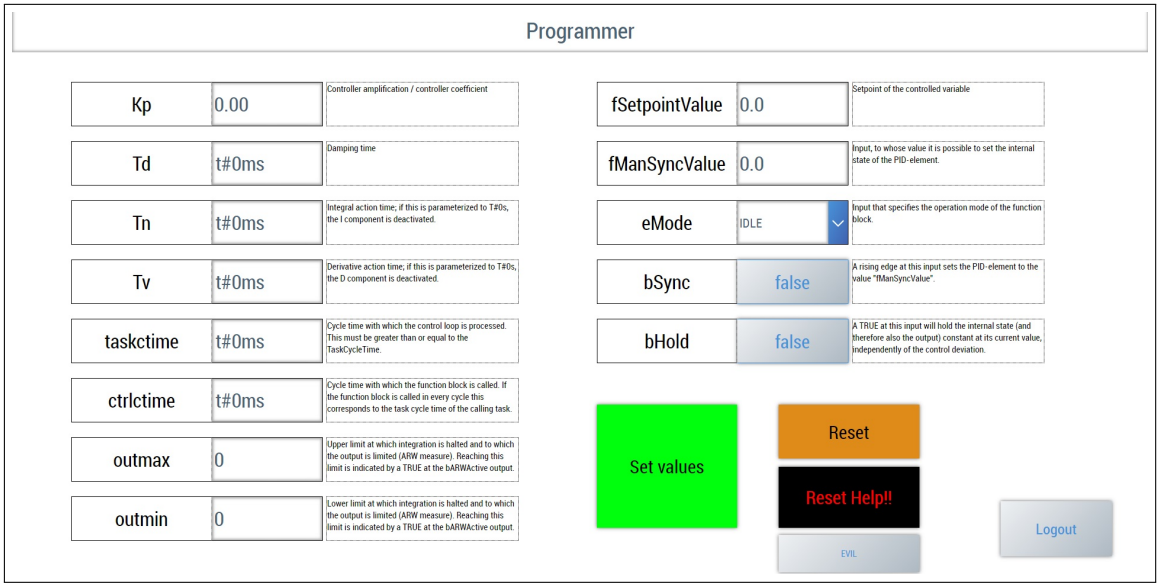

Figure 3.7: Programmer's Human Machine Interface

When the emergency button is pressed, or the temperature/pressure rise to values above their limit, an emergency occurs, popping up the emergency HMI. In the 3.8 Figure, a sitaution where the emergency button is pressed, therefore the text is red. When depressed, it will turn white and the X button appears in the right upper corner.

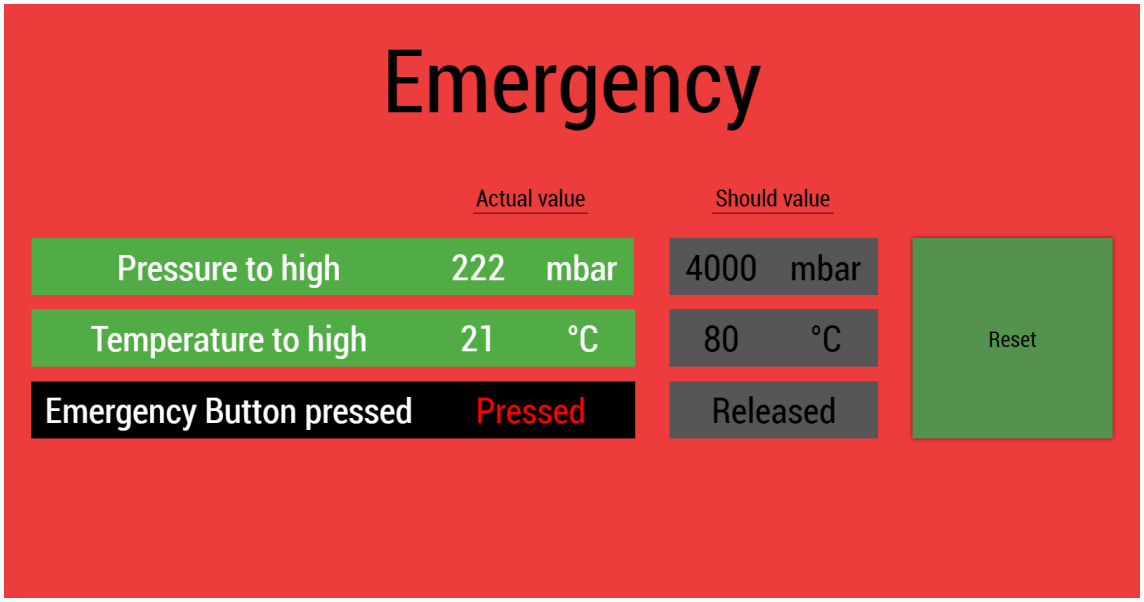

Figure 3.8: Emergency Human Machine Interface

The supervisor HMI (Figure 3.9), acceesible throguh '3333' pincode, represents an easy way to cntrol all of the important system function, such as overwriting the values for opening the vales and the motor and offering a complete overview over the transmitters data.

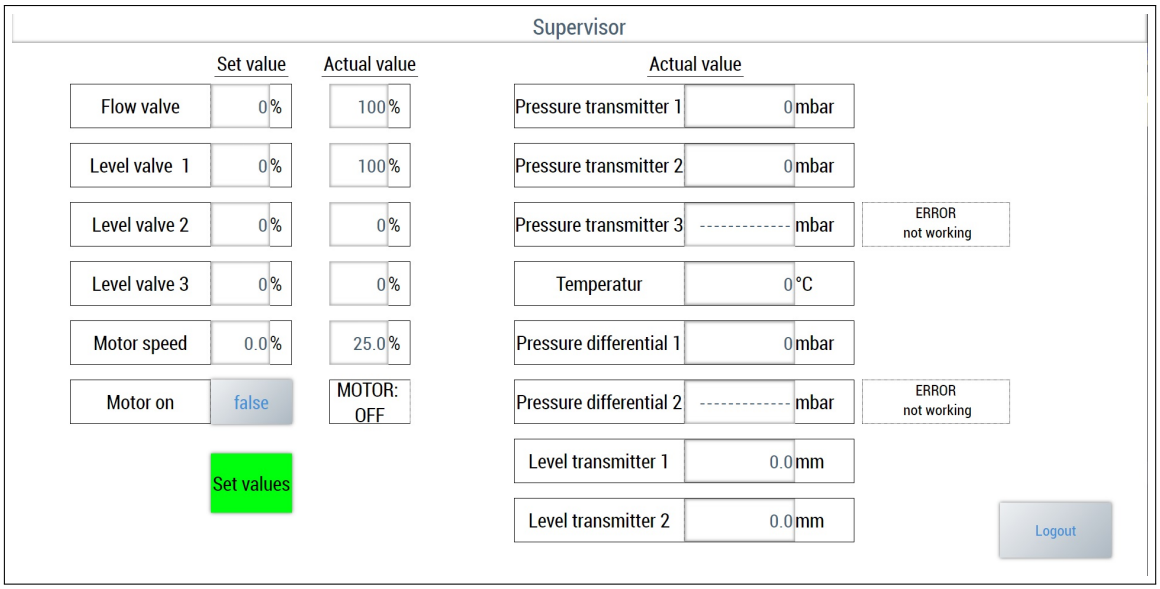

Figure 3.9: Supervisor's Human Machine Interface

3.4 Internal Acceptance Test

IAT (Installation Acceptance Testing) considers checking if everything's set up right at the actual place where machines work. It makes sure the hardware and software fit together correctly. If there are any issues during installation, IAT helps find and fix them. Passing IAT means the system is ready for real work.

3.5 Factory Acceptance Test

FAT (Factory Acceptance Testing) happens before the system goes to the real place. It represents a big check at the factory to make sure the control system meets the design plans. FAT catches and fixes any problems before the system gets to where it needs to be. Doing both IAT and FAT tests is super important for making sure everything runs smoothly.

The detailed reports for IAT and FAT tests are attached at the end of the raport for your look.

4 Conclusion

4.1 Challenges and Solutions

One of the challenges faced during the development process was working with the industrial principles of structured programming. This involved logical structuring the code into well-organized, modular and easy to maintain structures and an intuitive main program. To overcome this challenge, we adopted a structured programming approach, utilizing multiple sets of variables that we considered most relevant to be grouped together. If-else statements, loops and functions were utilized all throughout the code to facilitate the modularity character and ease of understanding, together with maintainability, making future changes and debugging easier.

Another challenge was optimizing the PID controller parameters in order to achieve the desired system performance and a minimum of 70% of oil extracted. This involved a combination of theoretical analysis with several experimental observations, fine-tuning the parameters until an optimal performance was achieved.

4.2 Future enhancements

The oil separator control project can be further improved by additionally controlling multiple parameters, such as further optimizations of the PID controller and more accurate timing to achieve even more precise oil separation. Furthermore, implementing advanced control algorithms, such as fuzzy logic or neural networks, could improve system adaptability and robustness.

In the HMI view of the programmer is a button called "Evil". This button opens the web page from Beckhoff about the PID. But when the window is opened, it can not be closed again. This is one thing that can be optimized at the HMI. There is no value of the amount of oil separated because the pressure differential sensor 2 is broken. The HMI is programmed that it would show values if the sensor works.

4.3 Final conclusions

The development of the oil separator system software was a rewarding experience. Through carefully designing, testing and implementation, we have successfully created a system that effectively manages the oil separation process.

5 Bibliography

References

[1] Beckhoff. "Cx5240 embedded pc." (), [Online]. Available: https://www.beckhoff.com/en-en/products/ipc/embedded-pcs/cx5200-intel-atom-x/cx5240.html?.

[2] ABB. "266hsh and 266nsh gauge and absolute pressure transmitters." (), [Online]. Available: https://library.e.abb.com/public/6e8beccd94ac4285a6a65c5bac1325a2/DS_266HSH_NSH_EN_X-02_2023.pdf.

[3] ABB. "At500 magnetostrictive level transmitters." (), [Online]. Available: https://new.abb.com/products/measurement-products/level/magnetostrictive-level-transmitters/at500.

[4] ABB. "2600t series pressure transmitters." (), [Online]. Available: https://library.e.abb.com/public/6e899e08c167764bc1257b0c0054738d/OI_266HART-EN-E-03_2011.pdf.

[5] ABB. "Sensytemp tshy (hy r)." (), [Online]. Available: https://library.e.abb.com/public/2b52da4e7bf84c9ab5d81e8a7875d034/10_10_364_EN_E01.pdf.

[6] Beckhoff. "Tf4100 tc3 controller toolbox - pid control." (), [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tf4100_tc3_controller_toolbox/245434251.html&id=.

[7] Beckhoff. "How to make a connection between twincat and beckhoff." (), [Online]. Available: https://www.youtube.com/watch?v=F4-1yEEtbJA.

[8] M. F. Aftab, *Separator manual*, 2023.

[9] M. F. Aftab, *Project introduction document*, 2023.

[10] M. F. Aftab, *Mas247 industrial it - teacher's class*, 2023.

[11] Beckhoff. "Tf4100 tc3 controller toolbox - overview." (), [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tf4100_tc3_controller_toolbox/9007199500176779.html&id.

[12] Beckhoff. "Tf4100 tc3 controller toolbox - definition of the structures and enums." (), [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tf4100_tc3_controller_toolbox/245508619.html&id=.

[13] "Manometer symbol." (), [Online]. Available: https://commons.wikimedia.org/wiki/File:Symbol_Manometer.svg.

[14] "Installed valve symbol." (), [Online]. Available: https://commons.wikimedia.org/wiki/File:Installed_Valve_Green.svg.

[15] Beckhoff. "Tcplclib tc2 utilities - fb$_b$*asicpid*." (), [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclib_tc2_utilities/34976395.html&id=.

6 Appendix

6.1 Simulation Function Block

```
function y = fcn(u, p_factor)

    if u > 100/6
        output = 0;
    elseif u >= 0
        output = 100-u*6;
    else
        output = 100;
    end

    if u >= 0
        input = 70;
    elseif u > -(30/18)
        input = (70+u*18);
    else
        input = 100;
    end
    y = input - output*p_factor;
end
```

Figure 6.1: Function block for Valve Control

6.2 Separator Drawing

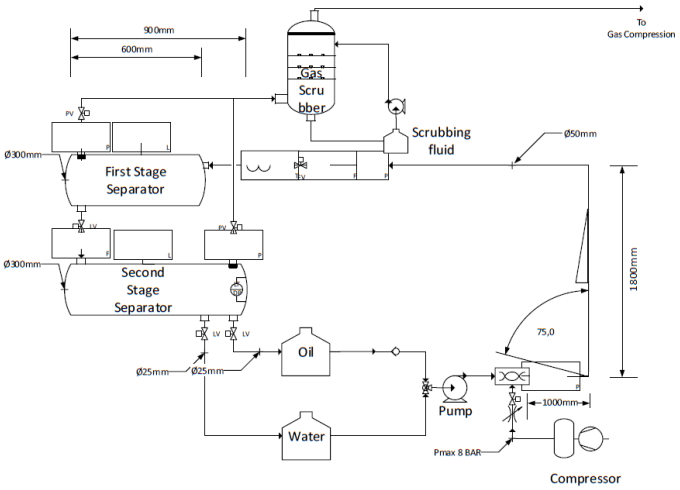

Figure 6.2: Separator Drawing

6.3 Gantt's chart

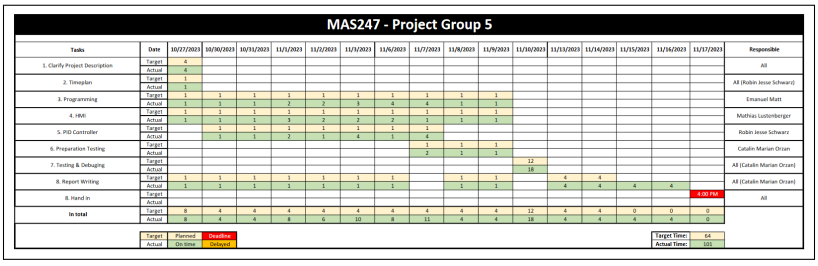

Figure 6.3: Group's Timeplan

UNIVERSITY OF AGDER
FACULTY OF ENGINEERING AND SCIENCE

Three-phase Separation Process Module

MAS247 Industrial IT Student Project, Fall 2023

Mathias Lustenberger, Emanuel Matt,
Catalin Marian Orzan and Robin Jesse Schwarz

Notes:

- Purpose: To verify that the "Three-phase Separation Process Module" meets the requirements of the Functional Design Specification.
- Scope: All module functions.
- Schedule: 2023-11-15
- Results: All test cases passed.
- Outstanding issues: Efficient separation.

Doc. Status:

	1.	Accepted
X	2.	Accepted w. minor comments. Work may proceed
	3.	Rejected
	4.	For information

Rev.No.	Date	Reason for Issue	Prep.	Chkd.	App.
1	15/11/2023	Emergency button delay	15/11/2023	15/11/2023	15/11/2023

Customer:	Document Title:			
UiA	**IAT/FAT** **THREE PHASE SEPARATOR**			
P.O. No.: 123456				
Customer Doc. No.: 7890	Doc. family: DOC	No. of pages: 6	Doc. No.: 123456	Rev.: 1

IAT/FAT

1 INTRODUCTION

1.1 GENERAL

This document shall verify that xxxxx works according to the specification given in document Functional Design Specification.

1.2 DEFINITIONS

SMDL	Projects document list (Supplier Master Document List).
EPMS	Projects activity plan (Engineering, Procurement and Manufacturing Schedule).
IAT	Internal acceptance test.
FAT	Factory acceptance test.
SAT	Site acceptance test.
FDS	Functional Design Specification.

1.3 REFERENCES

Functional Design Specification.

1.4 PUNCH TYPES DEFINITION

In the punch reporting, errors and diviations is required to be categorized. This will be used as a basis for authentication and non-approval criteria of the tests.

The following categories are used for errors and discrepancies:

Category A: Error in one or more features that causes the stop. For example, machine / system / function must be restarted.

Category B: Problems have serious consequences in the system or adjacent systems, such as errors in databases, serious errors in the reports so that they are incorrecly interpreted, etc.

Category C: Error with less serious consequences, such as smaller degree of instability, minor functions or propperty deficiency, etc.

Category D: Error without serious consequences, such as layout frailty or misprint in screens, reports or documentation that does not have consequences for the understanding of these.

The test is considered approved when 100% of the planned tests are carried out and there is no open A and B errors. In addition, a documented agreement on the further processing of the C and D errors are required.

1.5 PREPERATIONS BEFORE TEST

All of the physcal features were tested. With the remark of having PDT2 (Differential Pressure Transmitter) and PT3 (Pressure Transmit-
ter) physically faulty and the LV2 (Level Valve) partially clogged, the document was accepted with a minor fault, as the system can function without the sensor, but efficiency factor is affected.

2 TEST: EMERGENCY STOP AND SYSTEM HALT

2.1 TEST 1

2.1.1 EMERGENCY STOP AND SYSTEM HALT

Test Description:

- **Objective:** To verify the emergency stop functionality and immediate system halt.
- **Procedure:**
    1. Press the emergency stop button during system operation.
    2. Observe whether the motor stops and all valves close instantly.
- **Expected Result:** The system should come to an immediate halt with the motor stopped and all valves closed.

Test Results:

- **Initial Observations:** The emergency stop button was pressed during normal system operation.
- **Outcome:** The motor stopped abruptly, and all valves closed as expected.
- **Comments:** The emergency stop functionality performed as intended, providing a quick and effective means to halt the system in case of an emergency.

Test Revision:

- **Observations:** During subsequent tests, it was noted that the emergency stop functionality occasionally had a delay in response.
- **Action Taken:** The code was reviewed, and it was identified that a function call related to valve closure was placed above the emergency stop function in the execution sequence.
- **Resolution:** The code was revised to ensure that the emergency stop function takes precedence over other functions. The emergency stop response time was then retested, and the delay issue was resolved.

Verification:

- **Re-test:** The emergency stop button was pressed multiple times during subsequent tests.
- **Outcome:** The emergency stop function consistently halted the system promptly without any delays.
- **Conclusion:** The code revision successfully addressed the initial delay issue, and the emergency stop functionality now operates reliably.

2.2 TEST 2

2.2.1 PRESSURE AND TEMPERATURE ALARMS

Test Description:

- **Objective:** To validate the response of the system to high pressure and temperature conditions.
- **Procedure:**
    1. Simulate high pressure by setting PT1 and PT2 above 4 bar.
    2. Exceed the temperature limit by setting TT above 80^{o}C.
- **Expected Result:** Alarms should be triggered, stopping the motor and opening all valves to release pressure in case of high temperature.

Test Results:

- **Observations:** High pressure and temperature conditions were simulated as per the test procedure.
- **Outcome:** Alarms were triggered promptly, and the motor stopped while all valves opened to release pressure.
- **Comments:** The system responded appropriately to abnormal pressure and temperature conditions, ensuring the safety of the equipment and operators.

Test Revision:

- **Observations:** No issues were identified during the initial test.
- **Action Taken:** No code revision was required.
- **Conclusion:** The system's response to pressure and temperature alarms met the specified requirements without any need for revision.

2.3 TEST 3

2.3.1 LEVEL CONTROL AND BUFFER TANK FILLING

Test Description:

- **Objective:** To confirm the proper functioning of level control and buffer tank filling.
- **Procedure:**
    1. Fill LT1 and LT2 with liquid to simulate non-empty tanks.
    2. Initiate the program and observe the draining process.
- **Expected Result:** The draining process should occur if any of the tanks are not empty.

Test Results:

- **Observations:** LT1 and LT2 were filled with liquid to simulate non-empty tanks.

- **Outcome:** The draining process initiated correctly, indicating the proper functioning of the level control and buffer tank filling.
- **Comments:** The system correctly detected non-empty tanks and initiated the draining process as expected.

Test Revision:

- **Observations:** No issues were identified during the initial test.
- **Action Taken:** No code revision was required.
- **Conclusion:** The system's level control and buffer tank filling functionalities met the specified requirements without any need for revision.

2.4 TEST 4

2.4.1 FOAM HANDLING AND OIL SEPARATION EFFICIENCY

Test Description:

- **Objective:** To assess the system's ability to handle foam and achieve efficient oil separation.
- **Procedure:**
    1. Introduce foam into the system during operation by fast mixing.
    2. Monitor the system's response to foam and its impact on the separation process.
- **Expected Result:** The system should adapt to the presence of foam and still achieve effective oil separation.

Test Results:

- **Observations:** Foam was introduced into the system as per the test procedure.
- **Outcome:** The system functioned but the foam separation process was influenced by the foam.
- **Comments:** Document was accepted, considering this a minor fault, as it is mostly a fault from the physical model of the system, therefore can not be modified.

		IAT date / sign.	FAT date / sign.	
Item	**Description**			**Comment**
2.1.1	Test 2.1.1: EMERGENCY STOP AND SYSTEM HALT	15/11/2023/ G5	16/11/2023/ G5	Approved
2.2.1	Test 2.1.2: PRESSURE AND TEMPERATURE ALARMS	15/11/2023/ G5	16/11/2023/ G5	Approved
2.3.1	Test 2.1.3: LEVEL CONTROL AND BUFFER TANK FILLING	15/11/2023/ G5	16/11/2023/ G5	Approved
2.4.1	Test 2.1.4: FOAM HANDLING AND OIL SEPARATION EFFICIENCY	15/11/2023/ G5	16/11/2023/ G5	Approved