MAS247

Semester Project

Group 4

OIL SEPARATOR

THOMAS LØNNE STIANSEN
JAN LAKSESVELA HAUGSTAD
ADRIAN MATHIAS LERVIK LING

SUPERVISOR
Muhammad Faisal Aftab

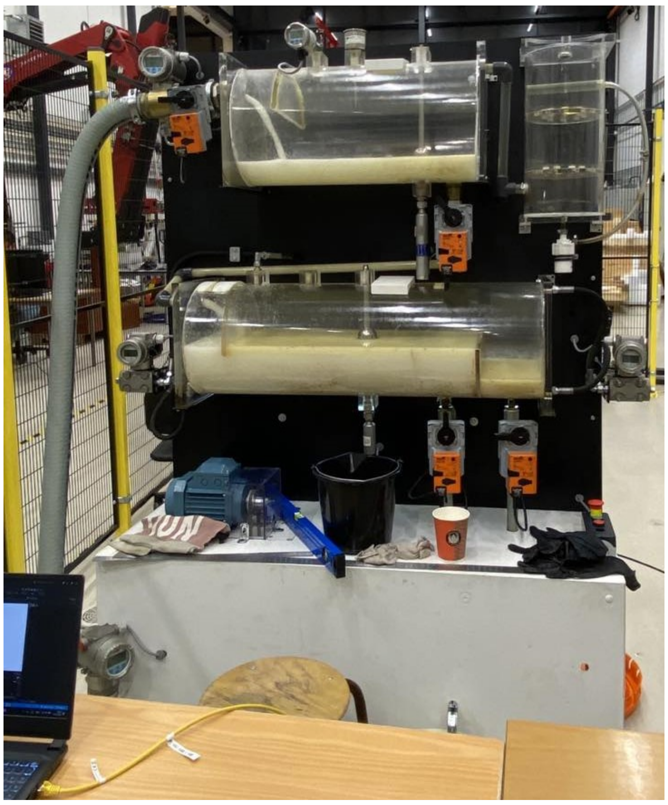

Figure 1: Oil Separator System

University of Agder, 2023
Faculty of Engineering and Science

Abstract

In this project, an oil separator is to be controlled with object-oriented programming of a programmable logic controller. It involves level transmitters, pressure sensors, differential pressure sensors and a temperature sensor. Signal conversion is central in this project, as the control logic is dependent on correct physical measurements. The objective is to take in a mixture of oil and water into a tank, send it down to the next tank and let the two liquids separate in order to let the oil flow over in a controlled manner into an oil container.

Contents

1 Introduction

Industrial processes require precise and robust systems, which measure real values and control them accordingly. This project represents the core of the oil and gas industry process control [1]. The use of gravity to let oil and water separate in tandem with PID-control is an example of how engineers can utilize control theory and signal processing to tune industrial processes. Tuning methods such as Ziegler-Nichols Open-loop tuning speed up the process, by estimating a starting point for the systems control.

2 Theory

Analog inputs and outputs

For analog signals, it is necessary to map the values that go in and out of the system to the corresponding values. Outputs are sent out of the system as bits to a Digital to Analog Converter (DAC), which sends them out as voltage in the range 2 [V] to 10 [V] [2]. The inputs from the sensors are sent into the system in the current range 4 [mA] to 20 [mA], which then are sent as bits into the system [2]. Both the inputs and outputs are using 16 bits, but only 15 for numbers; the 16th bit is for the sign (+/-).

Programmable Logic Controller (PLC)

In harsh conditions, such as the oil and gas industry, reliable and robust systems are a must, which requires a specialized computer with the same qualifications. This is where programmable logic controllers come in. They are simple to implement, and gives good control of inputs, outputs and the control that happens in between. They are used for real time operations, where precise timing is needed. It is common for them have the option to change the timestep of how quickly it updates. PLCs also have a long lifespan, compared to other options like microcontrollers. These controllers are very scalable, as the physical connections are modular, and they are programmable, which means the control logic can be expanded upon and modified after installation. The PLC can be controlled directly with a computer which is doing the logic, where the computer acts as a remote I/O system (RIO). The program can also be cross compiled onto the PLC, so the logic is run on the actual PLC.

Human Machine Interface

In order to control the systems in the real world, a interface between the PLC and the person using it is needed. When programming the PLC, the developer can add multiple specialized HMI screens.

Emergency System

In industrial processes, it is crucial to have emergency systems so no human can come in harms way. Such an emergency control must shut down the system until safe conditions are met. Usually they are implemented accompanied by a display system which informs the person supervising the process, and a clear button to disable the alarm once the conditions are back to normal. To shut down the system, there are multiple solutions. It is normal to have an emergency system running on a separate module, which interrupts the system, effectively shutting everything down in the moment. This solution is shown as the yellow module connected in figure 2.1.

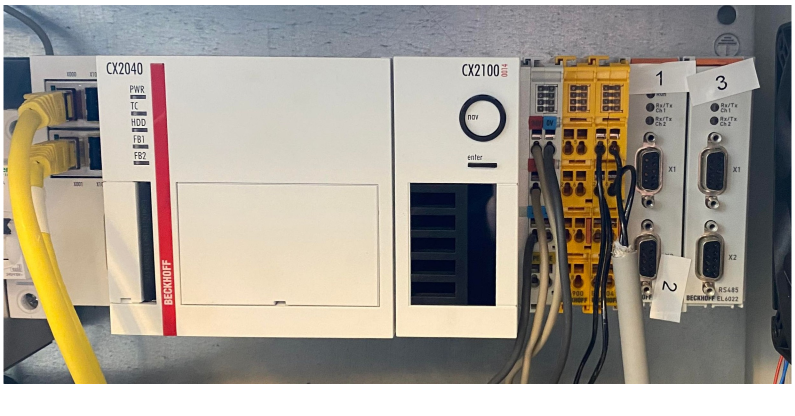

Figure 2.1: Emergency Module for PLC, connected to a robotic arm in the machine hall at UiA Grimstad.

Height Calculations

The level of oil and water in the tank can be modeled with a combination of the level transmitter (LT2) and differential pressure transmitter (PDT1). This can be done using an equation for each sensor, and solving for each height:

$$LT2 = h_w + h_o \tag{2.1}$$

$$\begin{aligned} PDT1 = \Delta P &= (P_w + P_o + P_{atm}) - P_{atm} \\ &= \rho_w \cdot g \cdot h_w + \rho_o \cdot g \cdot h_o \end{aligned} \tag{2.2}$$

Solving these equations give the following result:

$$h_w = PDT1 \cdot \frac{1}{g \cdot (\rho_w - \rho_o)} - LT2 \cdot \frac{\rho_o}{\rho_w - \rho_o} \tag{2.3}$$

$$h_o = LT2 - h_w \tag{2.4}$$

Where the symbols represent the following:

h_w	= height of water	$[m]$
h_o	= height of oil	$[m]$
$LT2$	= total liquid in tank 2	$[m]$
$PDT1$	= differential pressure in tank 2	$\left[\frac{kg}{m \cdot s^2}\right] = [Pa]$
P_w	= pressure from water	$\left[\frac{kg}{m \cdot s^2}\right] = [Pa]$
P_o	= pressure from oil	$\left[\frac{kg}{m \cdot s^2}\right] = [Pa]$
P_{atm}	= pressure from atmosphere	$\left[\frac{kg}{m \cdot s^2}\right] = [Pa]$
ρ_w	= density of water	$\left[\frac{kg}{m^3}\right]$
ρ_o	= density of oil	$\left[\frac{kg}{m^3}\right]$
g	= gravity	$\left[\frac{m}{s^2}\right]$

Where the constants are:

$$g = 9.80665 \left[\frac{kg}{m^3}\right] \qquad \rho_w = 998 \left[\frac{kg}{m^3}\right] \qquad \rho_o = 857 \left[\frac{kg}{m^3}\right]$$

And the two sensors LT2 and PDT1 are considered known, as they can be measured. The oil being used for the project is a high performance hydraulic oil with density according to the ISO VG 32 standard [5].

PID-controller A PID-controller can be used to take in the difference between desired value and actual value, where it takes the sum of the direct difference, the integral of it and the derivative of it. It can have be expressed in the s-domain:

$$G_{PID}(s) = K_P \cdot \left(1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s}\right) \tag{2.5}$$

It can also be expressed in the time domain:

$$G_{PID}(t) = K_P \cdot e(t) + K_I \cdot \int_0^t e(t)dt + K_D \cdot \frac{de(t)}{dt} \tag{2.6}$$

2.1 Mathematical Model: Tank Systems

Given a system of multiple tanks, one can find the differential equations to model the height of the liquids. This can be done using mass balance equations, where one considers the flow in to be equal the flow out. For two tanks, this can be expressed on the following form:

$$A_1 \cdot \dot{h_1} = Q - k_1 \cdot h_1 \tag{2.7}$$

$$A_2 \cdot \dot{h_2} = k_1 \cdot h_1 - k_2 \cdot h_2 \tag{2.8}$$

A circular tank, such as the ones for this system, can be modeled as a circular cylinder where the variable width can be found as a function of the height using the formula for a cylinder with a vertical offset equal to the radius of the circle:

$$\left(\frac{(z-R)}{R}\right)^2 + \left(\frac{hY}{R}\right)^2 = 1 \tag{2.9}$$

$$\Rightarrow hY = \pm\sqrt{\left(1 - \left(\frac{(z-R)}{R}\right)^2\right) \cdot R^2}$$

Using this, the variable horizontal area in the XY-plane is equal to the product of double the positive root of this expression and the length of the tank.

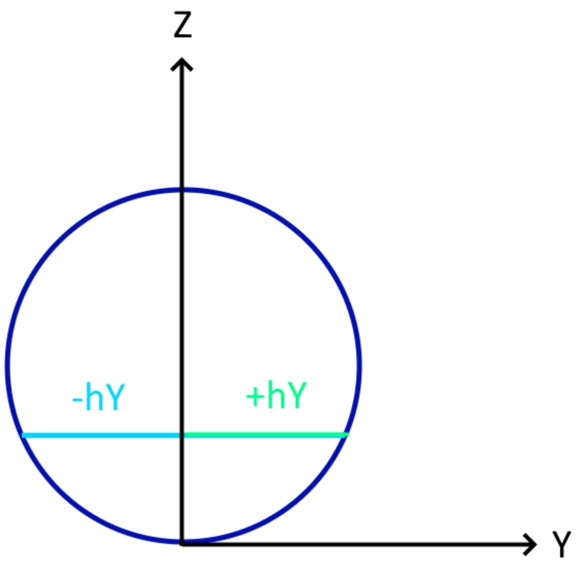

Figure 2.2: Cross section for variable area.

3 Method

The project was handed out along with a skeleton project with some of the most basic functions, such as mappings for the outputs and a starting point for the HMI. Throughout the project, version control in the form of git has been used. Before doing the project, a flow chart was designed to give general guidelines for the design process. This is shown in A.2. Additionally, milestones were done throughout the designprocess, which were noted in the IAT document, which is uploaded together with the project files.

3.1 Output mapping

To connect the variables in the program to the desired output on the PLC, they were only digitally bound to the corresponding etherCAT connections for EK1200, as the physical connection was provided.

Before implementing any kind of logic, the first objective was to get the inputs and outputs configured correctly. The analog output signals were already mapped, but the group verified the values. They take in the values in percent from the system, and use the following equation before sending the values out in bit:

$$X_{bit} = [\%] \cdot \frac{\Delta[bit]}{\Delta[\%]} + C[bit] \tag{3.1}$$

3.2 Button Mapping and Emergency System

Next the buttons were remapped. The goal was to have three physical buttons and their digital counterpart. The emergency "button" (tall red switch) and clear/reset button (red) are normally closed, while the toggle button (green) is normally open. Latches were implemented for the emergency switch to avoid it triggering multiple times in the loop. As for the toggle button, a debouncing logic was added. The emergency system was added with two functions; alarm handling with event logger library, and the logic for controlling the physical system. Since the project utilizes a PLC without a separate emergency module, the group compensated by integrating the physical control for the emergency system inside the "ControlSystem" POU. The alarm handling was done inside a function block, in order to keep it organized.

3.3 Sensor Readings and Calibration

The sensors were bound to EK1200 (such as the outputs) to the input variables inside of the main program. Most of them were scaled back from bits to desired values using the following equations:

$$\begin{aligned} X_{des} &= \frac{X_{bit}}{\Delta[bit]} \cdot \Delta[desiredUnit] + C[desiredUnit] \qquad (3.2)\\ &= \frac{X_{bit}}{(2^{15}-1)-(0)} \cdot \Delta[desiredUnit] + C[desiredUnit] \end{aligned}$$

where $\Delta[desiredUnit]$ was selected as the difference between the maximum and minimum value from the respective datasheets and physical specifications on the sensors. Regarding the two most crucial sensors, the level transmitters (LT1 and LT2) and differential pressure transmitter for the second tank's main container (PDT1); a more conservative approach was done. To achieve the correct desired output, the real values were measured, and the bits were logged. Using these values, linear regression gave the relation between the real values and the bit readings. For the level transmitters, the height was measured in addition to bit readings at a set of points. For PDT1, the tanks were emptied and the second tank was closed. Then clean water was added to the tank; height and bit readings were once again done intermittently. To empty the water out of the system, the group used a siphon, so no water was added to the system.

3.4 Control System

3.4.1 Program Structure

After the fundamental functions were implemented, the control system was designed as a separate POU to keep things organized. The inputs and outputs were set up to be scaled/mapped in the main POU (Program Organization Unit), so it updates the values continuously. The control system consists of three branches; the reset branch, the automated control branch and the emergency branch. Internal variables for the POUs are available to read from each other, but writing to them requires them to be listed as input variables. In the control branch, both manualMode and emergency will trigger a transition, effectively functioning as an interrupt, triggering in the moment of it happening and stopping the automated control.

3.4.2 Control Algorithm

The automated branch of the "ControlSystem" POU consists of three main parts; the filling step, the hysteresis controller for holding the level of the first tank constant, and the PID-controller for controlling the overflow of the liquid in the second tank. The former was implemented as one step connected in series with the two latter inside another step. The PID was added to the system using function blocks and structures from the controller toolbox. The input for the PID-controller was set to be the calculated water height, which was used to control LV1, while keeping LV2 fully open, considered a constant leakage. Ziegler-Nichols tuning for the PID-controller was done with a step input for LV1 and measuring the output height from LT2.

3.5 Human Machine Interface

The human machine interface was separated into three separate screens: engineering view, operator view and trend view. The engineering screen is for the engineers who know how the system works, so they can change the parameters such as the setpoint for the PID-controller and the valves while working on the system. The operator screen is for operators without proper knowledge using the system, and only serves the most basic functions, such as buttons for starting and stopping the process, and buttons for emergency trigger and clearing. The trend view displays the valve output signals in percent, and the remapped sensors.

3.6 Mathematical Models

The mathematical models for each tank were expressed with equations 2.7 and 2.8. In order to simulate the models, Q, k_1 and k_2 need to be found. The motor was set to a constant percent, where the height of the tank was measured, where the change of volume could be related to the flow into the system Q. In order to find k_1, the flow of the motor was set to the same constant percent, where the height was measured when the tank was in equilibrium and the water level was approximately constant; ie. flow in was equal to flow out. Finally, the last coefficient k_2 was found by the relation between the valve areas. To find these, the diameter was measured with calipers. When the differential equations' parameters were found, the mathematical model was simulated in MATLAB and plotted with the measured values.

4 Results

4.1 Output Mapping

The mapping from equation 3.1 have the following values:

$$\begin{aligned} X_{bit} &= X_{\%} \cdot \frac{BitMax - BitMin}{PercentMax - PercentMin} + BitMin \\ &= X_{\%} \cdot \frac{(2^{15}-1)\cdot(1-0.2)}{100-0} + 0.2 \cdot (2^{15}-1) \end{aligned}$$

The reason for 20% of the maximum value for the bit range is that it has the same scaling as the output in Voltage:

$$Ratio = \frac{V_{min}}{V_{max}} = \frac{2[V]}{10[V]} = 0.2$$

4.2 Button Mapping and Emergency System

The toggle button debouncing logic was implemented using the rising edge detector from Twin-CAT's standard library, see A.1.1. That resulted with the button only changing the state on rising edge of the signal.

In order to have the emergency system logic interrupt the main POU, a parallel transition was added to each of the control steps, so it would leave as soon as the alarm/emergency triggered. The emergency system was set up to trigger if the emergency switch (either digital or physical) was pressed, if the pressure went beyond 3 [bars] or if the temperature went beyond 60 [°]. While testing, the emergency system was set to run until a time had passed, but in the finalized version, it was set to clear once the levels went below 5 [cm] (plus the pressure and temperature going to safe levels, obviously).

4.3 Sensor Readings and Calibration

The mappings for the first sensors were done using the relations from 3.2, which gave the following results:

4.3.1 Pressure Transmitters (PT1 and PT2)

$$PT_{bar} = PT_{bit} \cdot \frac{maxPT - minPT}{bitMax - bitMin} + minPT = PT_{bit} \cdot \frac{(6)-(0.06)}{(2^{15}-1)-(0)} + 0.06 \tag{4.1}$$

where the range in [bar] was read from the physical sensor.

4.3.2 Temperature Transmitter (TT1)

$$TT_{^\circ C} = TT_{bit} \cdot \frac{maxC - minC}{bitMax - bitMin} + minC = TT_{bit} \cdot \frac{(250)-(-50)}{(2^{15}-1)-(0)} + (-50) \tag{4.2}$$

where the range in [$^\circ C$] was found in the datasheet [3].

4.3.3 Pressure Differential Transmitter (PDT2)

$$PDT1_{bar} = PDT1_{bit} \cdot \frac{maxPDT - minPDT}{bitMax - bitMin} + minPDT = PDT1_{bit} \cdot \frac{(0.4) - (0.004)}{(2^{15} - 1) - (0)} + 0.004 \quad (4.3)$$

where the range in [bar] was read from the physical sensor.

4.3.4 Level Transmitters (LT1 and LT2)

For these sensors, the bit readings and measured height was plotted in MATLAB. The slope was calculated using the following equation:

$$Slope_{LT1} = \frac{\Delta[mm]}{\Delta[seconds]} = \frac{9}{1694} \quad (4.4)$$

$$Slope_{LT2} = \frac{\Delta[mm]}{\Delta[seconds]} = \frac{9}{1529} \quad (4.5)$$

Then the offset was added as a constant:

$$C_{LT1} = 37[mm] \qquad C_{LT2} = 43.4[mm]$$

4.3.5 Pressure Differential Transmitter for second tank's main container (PDT1)

Finally, the differential pressure sensor was calibrated. To do this, the group filled the tank with only fresh water as a reference. The heights were measured at each interval, in addition to bit readings. The pressure was calculated for the height list. To empty the tank, a siphon was used, as shown in section A.3.1. The result was plotted in MATLAB with Pascal as a function of bits; where the "Curve Fitter Toolbox" was used to find the coefficients for the regression line:

$$\begin{aligned} PDT1_{Pa} &= PDT1_{bit} \cdot k + b \\ &= PDT1_{bit} \cdot 0.06172 + 69.67 \end{aligned}$$

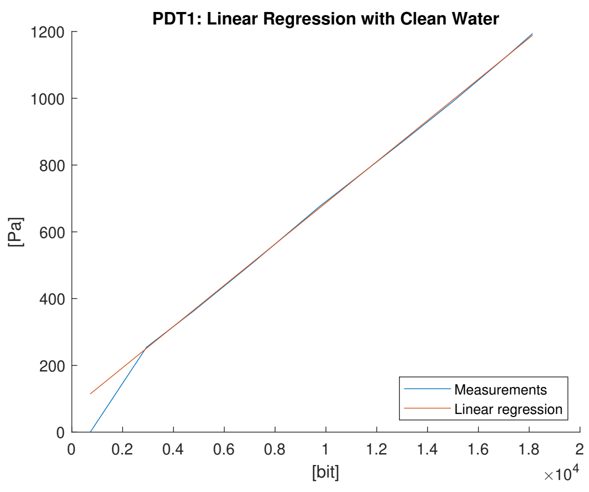

Figure 4.1: PDT1: Linear Regression comparison

4.4 Control System and Program Structure

The main POU was made in Structured Text (ST), and took care of mostly everything except the control logic. An initializer function block was made in order to reset the system's parameters in multiple places. This is used for the first scan in the main POU and also for when the reset branch of the ControlSystem POU triggers. The main POU also handles the "FB_EmergencySystem" function block, which is used for alarm handling and logging the emergencies. The alarm handling was done using the Tc3_EventLogger library, where an instance of its function block was created in a global variable list, to make it available everywhere. The emergency system function block was implemented with logic corresponding to the control system logic for when the alarm raises and clears, but also needed logic for initializing the event logging of the alarm. Since it is not possible to call functions inside the header, boolean logic was used to do it only once. The main POU takes care of the boundary conditions, so the outputs don't go outside the specified ranges [2]. Furthermore, it handles the rising edge detector for the toggle button, and also short logic for displaying its digital counterpart. Last but not least, the main POU calculates the height of the water and oil, using equations 2.3 and 2.4.

The ”ControlSystem” POU was done with a Sequential Function Chart (SFC), where the system got divided into the branches mentioned above. The emergency branch was set to turn off motor and open the valves on entry, and check continuously for clear/reset signal before the timer, so it would latch the state and clear once the ”safe” transition triggered. The reset branch utilizes the initializer function block mentioned previously, and also changes the alarmState for HMi display and ”pre-safe state clear latch”. The ”Filling” step of the control branch sets the valves and motor to a defined state until the level reaches about 4-5 [cm] below the weir. The hysteresis algorithm was implemented as a function block, so it could be called simultaneously with the PID-control inside the ”Controlling” step of the control branch. This resulted with the level of tank 1 being almost constant while PID-controlling the height. The hysteresis was set to turn the motor on when the level goes below 5 [cm] and turn off when it goes above 6 [cm]. The PID-controller was implemented, using the Tc2_ControllerToolbox library. The mode and PID parameters for the library's structure were set inside the ”Init” step, and then the mode was changed to active once the ”Controlling” step activated. While active, it calls the PID function block to update the values for the output to the valve depending on the measured and calculated water height. The speed of the motor was set to 20% for the process, despite the recommended 30-35% [2].

4.4.1 Ziegler Nichols tuning: Open Loop Step Response

The Ziegler Nichols open loop tuning was done with a step response from 0% to 100% input for LV1, and logging the output from LT2. The output was then scaled to percent for LT2 from bottom of tank (0%) to the overflow height (100%). Since this was done, also the inputs to the PID-controller had to be in percent, to make use of the ZN-tuning values. The scaling equation for both the ZN-tuning in MATLAB and PID-control in TwinCAT was done using the difference of the height in [mm]:

$$h_{\%} = h_{mm} \cdot \frac{max_{\%} - min_{\%}}{max_{mm} - min_{mm}} \tag{4.6}$$

$$= h_{mm} \cdot \frac{100 - 0}{156.84 - 43.4} \tag{4.7}$$

This resulted with the inputs for the PID-controller being in percent, as shown in figure 4.3.

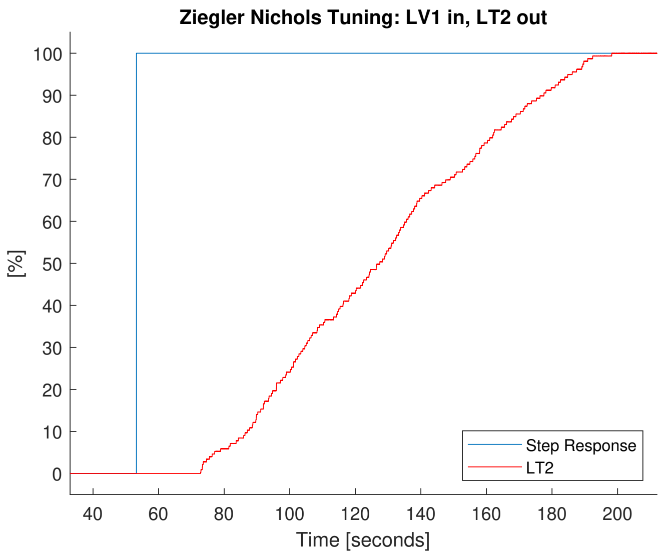

Figure 4.2: Step response of LV1 and LT2.

The tuning was done by manual reading, since a filter would have been needed to do it analytically as it had some noise in the signal which gives a steep and incorrect slope. The values from the Ziegler-Nichols tuning gave the following parameters:

$$K_P = 8.9318 \qquad T_n = 26740[ms] = T_i \qquad T_d = 6685[ms]$$

Relating them to the time domain coefficients:

$$K_P = 8.9318 \qquad K_I = 0.33402 \qquad K_D = 0.74845$$

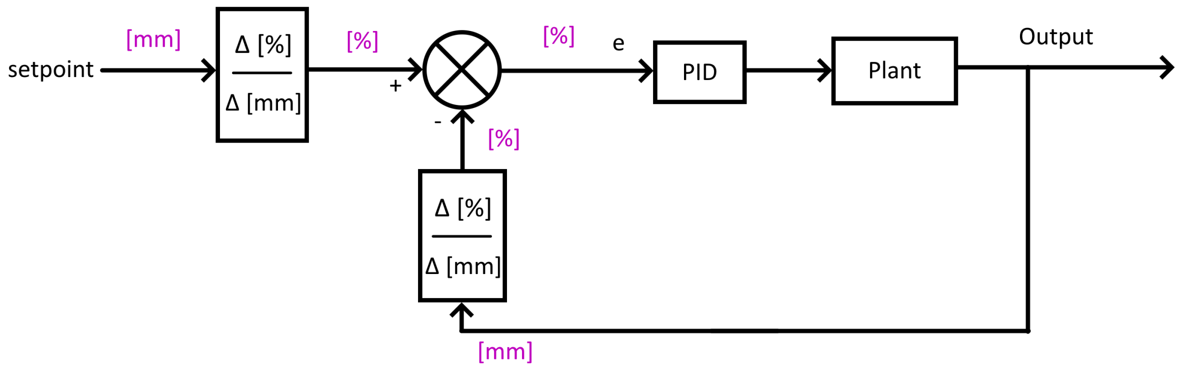

Figure 4.3: Unit Block Diagram

After running the system with a setpoint for the water height into the PID-controller as 84 [mm], the oil tank had separated the oil as shown in figure 4.4.

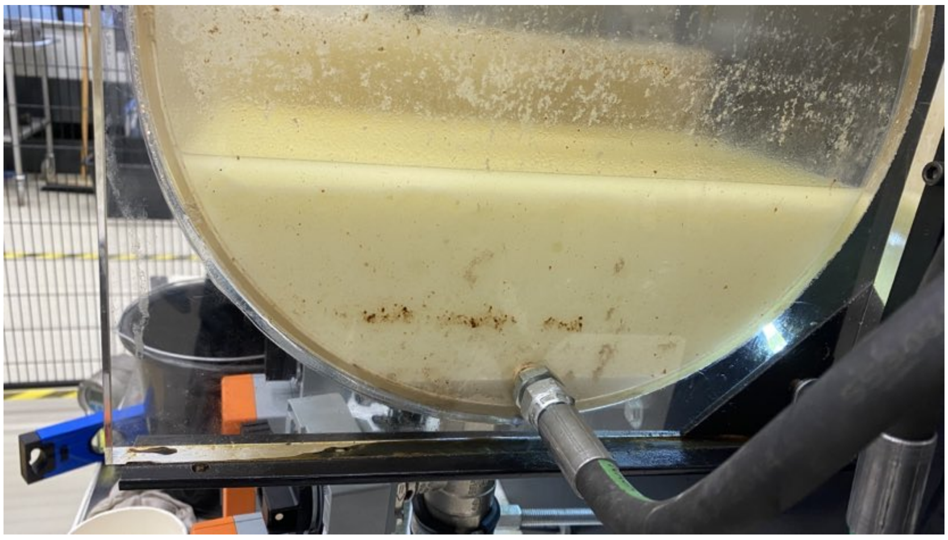

Figure 4.4: Oil tank after PID-control

4.5 Human Machine Interface

The engineering view uses ”Numeric Input” blocks in order to take in the values for the valve percentages and the motor speed, as well as the PID setpoint. Toggle buttons were used for turning the motor on and off, and switching between manual and automatic mode. The emergency system was implemented with three colors, indicating if it is either raised, cleared before safe state or cleared after safe state. In order to achieve this, an enum was created with the three states, and the triangles colors were bound to the instances using switch cases.

4.6 Mathematical Models

The pump flow rate Q was found experimentally by setting the motor to 20% speed and filling tank 1 to a height of 150 [mm]. By calculating the volume of the fluid at 150 [mm] and diving by time, the input flow rate is obtained.

$$Volume = L \cdot \left(\cos^{-1}\left(\frac{r-h}{r}\right) \cdot r^2 - (r-h) \cdot \sqrt{2 \cdot r \cdot h - h^2} \right) \tag{4.8}$$

Equation for area used with the length was found online [4].

$$Q = \frac{\Delta Volume}{\Delta Time} = \frac{20.3[L]}{60[s]} \approx 0.34\frac{[L]}{[s]} \tag{4.9}$$

Tank 1 was observed through visual confirmation and scope data to be in a steady state when the water height reached 42 [mm]. It can be ascertained that the flow through LV1 is equal to the input during this time. Assuming that the outflow is proportional to water height, the coefficient k_1 was found:

$$k_1 = \frac{Q}{h_1} = \frac{340\,000 \left[\frac{mm^3}{s}\right]}{42[mm]} \approx 8095 \left[\frac{mm^2}{s}\right] \tag{4.10}$$

The coefficient k_2 was found by taking the product of the coefficient and the ratio of the valve surfaces.

$$k_2 = k_1 \cdot \frac{A_{V2}}{A_{V1}} = 8095 \left[\frac{mm^2}{s}\right] \cdot \frac{\pi \cdot \left(\frac{8}{2}[mm]\right)^2}{\pi \cdot \left(\frac{21}{2}[mm]\right)^2} \approx 1175 \left[\frac{mm^2}{s}\right] \tag{4.11}$$

The resulting mathematical model from the differential equations 2.7 and 2.8 were plotted in MATLAB, together with the measured heights from the level transmitters. The simulated height was calculated dividing the tanks into small volumes, with respect to Δt using a for loop, see section A.1.1. This is shown in figure 4.5.

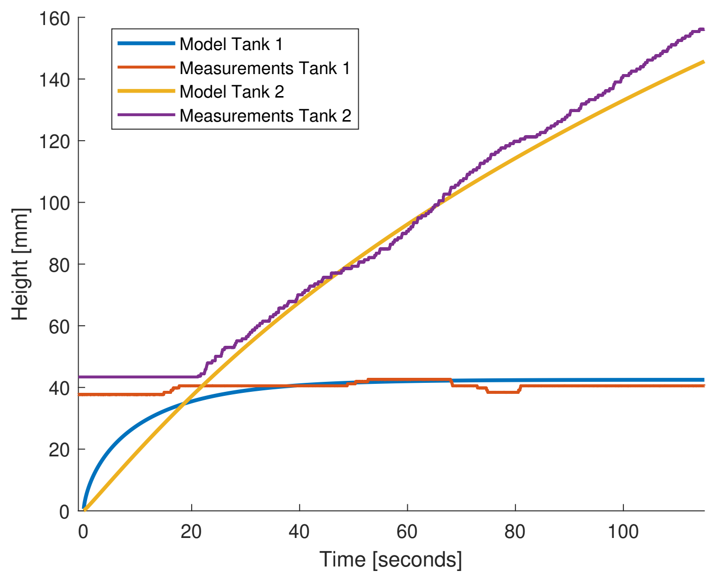

Figure 4.5: Mathetmatical Models compared to real values

5 Discussion

The use of version control has been great for the project, giving the opportunity to have backups, as well as the ability to revert to former versions.

5.1 Output Mapping

The mapping was done with the lowest output value being 2 [V], which was later corrected in the manual after the project had begun [2].

5.2 Button Mapping and Emergency System

The red and green buttons' functions were changed, to give more functionality to the system. This allowed the motor to be toggled on and off using the green button, while also allowing for clearing alarms and resetting the values using the red button. It was later realized that this meant the operator would not have a separate button for turning on and off the automated control. If there would have been given more time, this would have been fixed. A solution could have been to have the green button toggle the automated control instead of the motor by binding them to the "manualMode" variable. The emergency system was implemented and checked for the button and the temperature transmitter. Due to time limitations, the pressure logic didn't wasn't checked. The reason for this was that the pressure readings needed verification with a reference, as some of the sensors had a unknown gain and offset which made them not able to be scaled directly from the datasheets/physical specifications. The maximum pressure and temperature was specified to be 4 [bar] and 80 [°C], but the group added a safety factor and set them to be 3 [bar] and 60 [°C].

5.3 Sensor Readings and Calibration

As mentioned in section 5.2, some of the sensors had a unknown gain and offset. These could be changed physically by opening the sensors and changing them with buttons on them; but the group decided to leave the system as is. Therefore, linear regression was done to find the relation between desired output and actual output. The height for the level transmitters was fine by just measuring, as it is not dependent on the type of liquid it floats on. The rationale behind this and the equations 2.3 and 2.4 being based on the level transmitter floating on top of both liquids, is the density of the liquids. The density of both the water and the oil are quite close, and therefore it was a logical decision that the transmitter probably did not have a density between them. This was also verified as the measured height was correct throughout the physical tests, even when varying the amount of oil being mixed into the tank. The offset for the level transmitters were around the middle of them, as that is when the liquid has enough buoyancy to push them up. Most importantly, the differential pressure sensor for tank 2 needed linear regression to find the actual value. As the liquid had already been mixed *thoroughly* multiple times by all the groups doing the project, the mixture effectively had an unknown density. Therefore the siphoning was done, since water has a known denisty. This meant the group could find the slope and offset relating the actual pressure to the measured bits. Before doing the regression, the estimate of the water and oil's height was not accurate.

5.4 Control System and Program Structure

It was important when designing such a system to keep things organized. This makes it more effective to work on, and easier to isolate the source of errors. An object oriented approach made the project more clear and easier to read as well. In regards to error isolation, that was what spawned the water test to find the actual gain and offset of the pressure sensor. Prior to this test, the water and oil estimate was as mentioned above poor. Still, when using the new regression for the pressure, the estimate for the water height was not entirely as expected. The group expected the height of the water to be quite close to the weir, as there is a fair amount of water in the system compared to oil. The reason for the incorrect estimate is probably due to the liquids being mixed well, as mentioned in the previous section. Also, the fact that the liquids have relatively close densities, points towards the fact that they don't separate too well, compared to other oils with a bigger difference in density. Another mistake that was corrected was the use of units for the PID-tuning. The group originally tuned for percent and took in millimeters, which later was corrected. After correcting the PID-tuning, the values were compared to the time domain coefficients. This showed that the proportional term was substancially larger than the other coefficients, which was not the case prior to the change. That also pointed towards the values being more correct, as a dominating proportional term was expected.

A part of the project description was to have the goal to separate enough oil to read 70% from PDT2. This would have been added, but due to the pressure sensor being unavailable for the majority of the project, the group decided to discuss the situation instead. 100% would have been defined as the top of the weir, if implemented. The reason for 70% is most likely due to the fact that the density of the oil being less than water. If the tank has a set pressure and one compares the tank being filled with only water and only oil, it would yield different heights. The group concluded that the separated oil mixture shown in figure 4.4 was adequate, as the volume of the water is much lower than the oil, considering the walls of the cross section up to the weir acts as a parabola. In this small system, there will run some water over to the oil tank, since it does not have time to separate completely. Either having a bigger tank, longer time or higher difference in denisty could combat this.

5.5 Human Machine Interface

The group made three HMI views, which made it organized. If more time would have been provided, differential equations would have been simulated and plotted in the trend view simultaneously as the system running. A simple implementation which would have also been added, was referances for the trend views, such as the height and pressure at the bottom and top of the tank, and also the weir.

5.6 Mathematical Models

There was a clear connection between the mathematical models and the actual measured values, as shown in figure 4.5. The main difference between the two, other than noise, is the initial values. The reason for this difference is the fact that the level transmitters does not move before the level has gone high enough to push them up with the buoyancy, which is around the middle of them.

6 Conclusion

The project was completed successfully, and gave the group a good introduction to programmable logic controllers and object-oriented programming. A lot was learned when it comes to isolating the issues, and being critical to sensor values, even when reading documentation. Even though the pressure sensor for the oil was not being used to control when the process was finished, the group was satisfied with the results. The project gave a really good introduction to how one can implement emergency systems, and how important it is with safety in such systems and industry. Furthermore, a good introduction to human machine interface design was given, where the students learned to set up the system for different purposes and groups of people.

A Appendix

A.1 Code

A.1.1 Button Debouncing Logic with rising edge detector

Debouncer Header

```
        // Toggle Motor Statee with green button
risingEdge : R_TRIG;
elapsedDebounce : TIME;
toggleMotor : BOOL;
```

Debouncer Script

```
        // Toggle motor state with green button
risingEdge(CLK := Toggle_Button,
           Q => toggleMotor);
IF toggleMotor THEN
    Motor_ON_OFF := NOT Motor_ON_OFF;
END_IF
```

Mathematical Model: Simulating Differential Equations with small volumes

```
for i = 1 : 12000
    % Area calculations - discrete volume integral
    tank1Areal = 1 + 2 * L1 * sqrt( ( 1 - ((tank1h - R)/R)^2 ) * (R^2) );
    tank2Areal = 1 + 2 * L2 * sqrt( ( 1 - ((tank2h - R)/R)^2 ) * (R^2) );

    tank1h = tank1h + deltaT*(tank1input - tank1k*tank1h)/tank1Areal;
    tank2h = tank2h + deltaT*(tank1k*tank1h - tank2k*tank2h)/tank2Areal;

    tank1Vec(i) = tank1h;
    tank2Vec(i) = tank2h;
end
```

A.2 Flow Chart

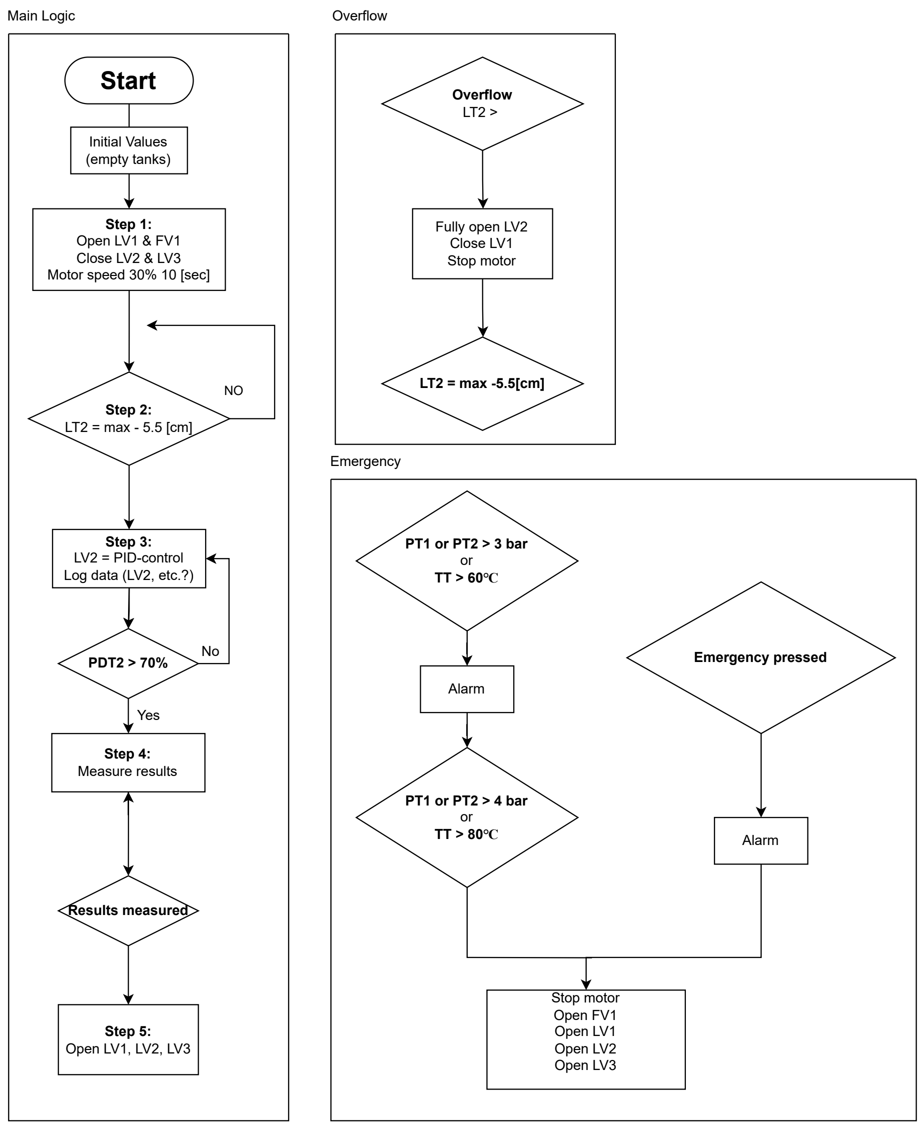
Main Logic
Start
Initial Values (empty tanks)
Step 1: Open LV1 & FV1 Close LV2 & LV3 Motor speed 30% 10 [sec]
Step 2: LT2 = max - 5.5 [cm]
NO
Step 3: LV2 = PID-control Log data (LV2, etc.?)
PDT2 > 70%
No
Yes
Step 4: Measure results
Results measured
Step 5: Open LV1, LV2, LV3
Overflow
Overflow LT2 >
Fully open LV2 Close LV1 Stop motor
LT2 = max -5.5[cm]
Emergency
PT1 or PT2 > 3 bar or TT > 60℃
Alarm
PT1 or PT2 > 4 bar or TT > 80℃
Emergency pressed
Alarm
Stop motor Open FV1 Open LV1 Open LV2 Open LV3

Figure A.1: Flow Chart Diagram

A.3 Images from the machine hall

A.3.1 Siphoning of tank 2

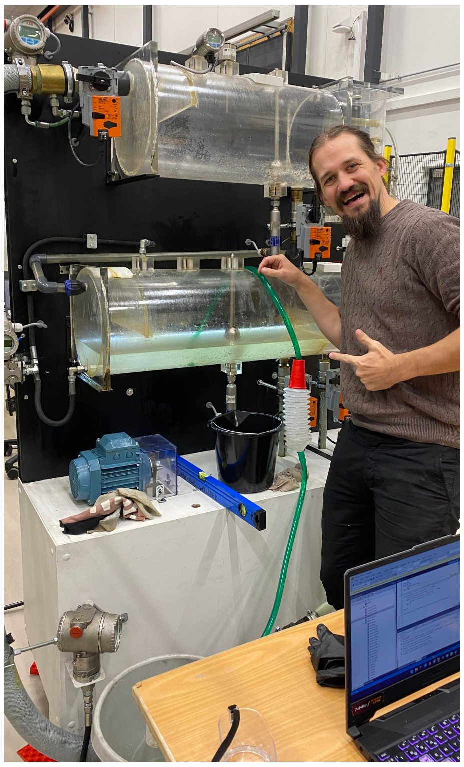
Figure A.2: Siphoning of second tank, for PDT1 regression.

Bibliografi

[1] Muhammad Faisal Aftab. *MAS-247 Industrial IT Project*. URL: https://uia.instructure.com/courses/13948/files/2358454?module_item_id=536481. (Accessed: 20.10.2023).

[2] Muhammad Faisal Aftab. *Three phase separator User Manual and Guidelines*. URL: https://uia.instructure.com/courses/13948/files/2358452?module_item_id=536482. (Accessed: 20.10.2023).

[3] ABB Measurement Analytics. *Resistance thermometer for the food and beverage industries*. URL: https://library.e.abb.com/public/2b52da4e7bf84c9ab5d81e8a7875d034/10_10_364_EN_E01.pdf. (Accessed: 10.11.2023).

[4] Rod Pierce. *Volume of Horizontal Cylinder*. URL: https://www.mathsisfun.com/geometry/cylinder-horizontal-volume.html. (Accessed: 16.11.2023).

[5] The Engineering ToolBox. *ISO Grade Oils - Viscosities and Densities*. URL: https://www.engineeringtoolbox.com/iso-grade-oil-d_1207.html. (Accessed: 13.11.2023).