Development of SCADA System to control and monitor the Temperature of Air Heater

R.P. Madusanka Niroshan Rajapaksha
University of South Eastern Norway
Porsgrunn, Norway
258954@usn.no

***Abstract*—Purpose of this study is to develop a Supervisory control and data acquisition (SCADA) System for Air Heater. This system includes a control system, data logger and alarm system. OPC UA server has been used to communicate between control system and the data logger. SQL database is used by the data logger to store the data. OPA server and control system have been developed by the LabVIEW, data logger has been developed by the C# in the visual studio and alarm system has been developed by the ASP. NET in the visual studio.**

Keywords—SCADA, Air Heater, Control System, Data Logger, Alarm System, OPC UA, SQL Server, LabVIEW, C#, ASP.NET

I. Introduction

Supervisory control and data acquisition (SCADA) can be identified as a computer based system which is used to collect and analyze real time data. Also, this data is used to control and monitor the physical equipment. Communication is the most important factor in SCADA. In this study SCADA system has been built to control and to monitor the temperature of air heater. It consists of a control system, a data logger and an alarm system. OPC UA server is used to communicate between control system and the data logger. SQL server is used to communicate between data logger and database. Communication between alarm system and database is also done by SQL server. Figure 1 shows the data flow between each system.

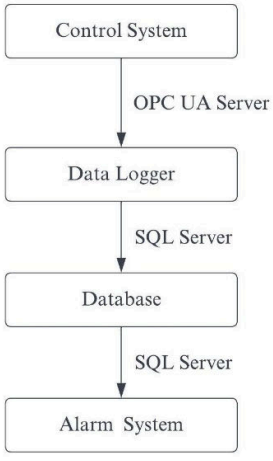


Figure 1: Simple illustration of data flow of SCADA system

II. Instruments

A. Air Heater

Main instrument of this study is air heater and it has inputs and outputs. Control voltage input and Temperature output are considered in this study. Figure 2 shows an image of the air heater.



Figure 2: Air Heater [1]

B. USB 6008

This instrument is used to communicate between PC and the air heater. It has analog inputs, analog outputs, digital inputs and digital outputs. Figure 3 shows an image of the instrument.



Figure 3: USB 6008 Instrument [1]

III. Control System

A. LabVIEW Program

Purpose of this is to control the output temperature of air heater with respect to the reference temperature. This program has a simulator which can be used to experience the operation of air heater without making physical connections. Also, it is possible to connect real air heater and control it. Figure 4 shows the login page, figure 5 shows the controlling page and figure 6 shows the information page.

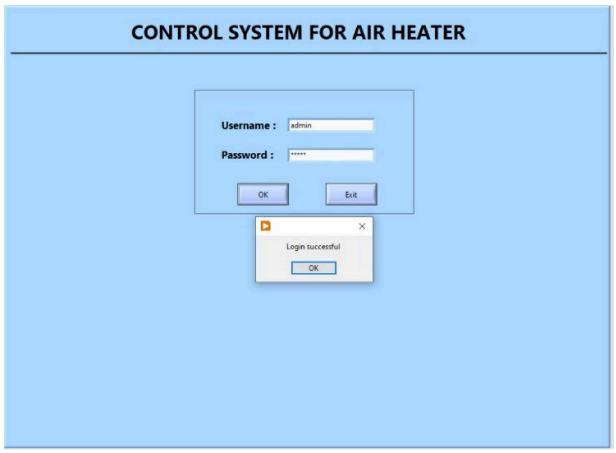


Figure 4 : Login page of LabVIEW program

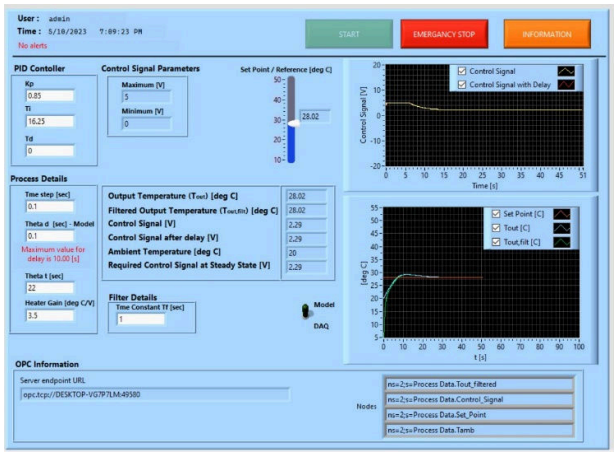


Figure 5: Control page of the LabVIEW Program showing the set point tracking – After tuning PID controller

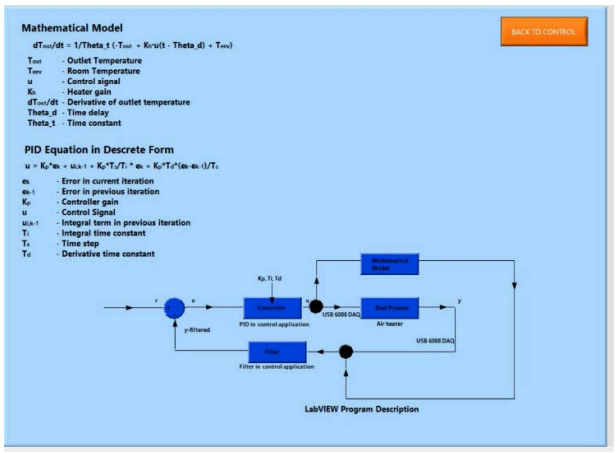


Figure 6 : Information page of the LabVIEW Program

LabVIEW program has been designed with state machine and there are 6 states in the program and block diagrams (except stop state) are shown in figures 7, 8, 9, 10, 11, 12.

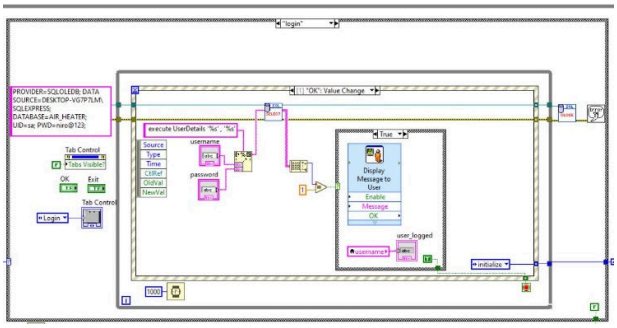


Figure 7 : Block diagram of the Login state

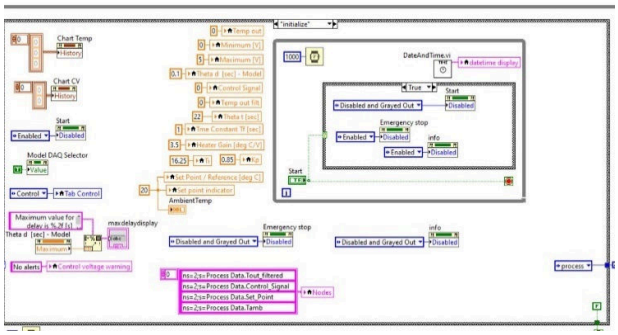


Figure 8: State for assigning initial values

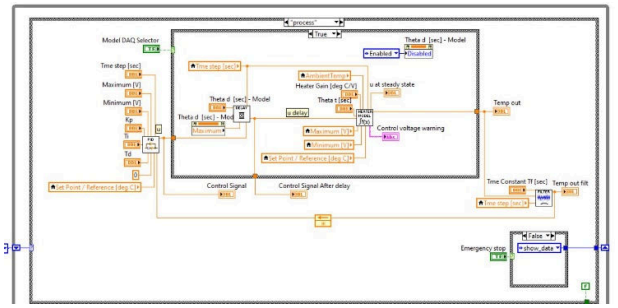


Figure 9: State for the main process – Mathematical Model

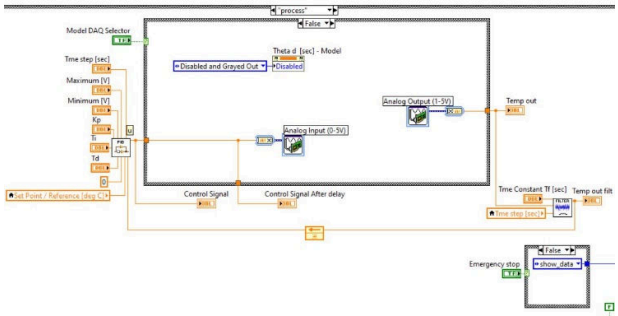


Figure 10: State for the main process – with DAQ

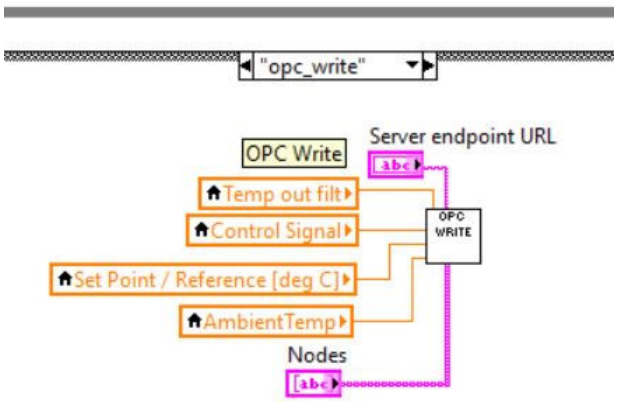


Figure 11: State to write data on OPC server

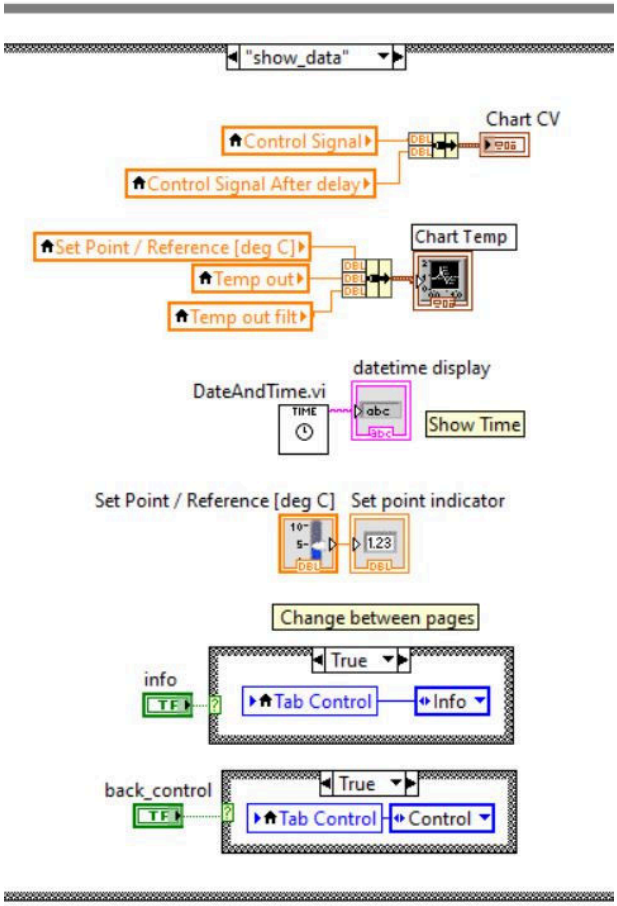


Figure 12: State to show data on plots, show time and change between pages

This LabVIEW program consist of 6 Sub VIs for followings tasks.

- PID Controller (Figure 13)
- Delay Function (Figure 14)
- Low Pass Filter (Figure 15)
- Date and Time Creation (Figure 16)
- Air Heater Mathematical Model (Figure 17)
- OPC Writer (Figure 18)

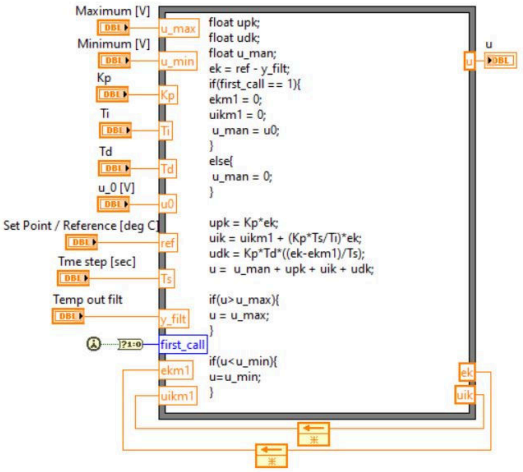


Figure 13 : Sub VI for PID controller discrete operation

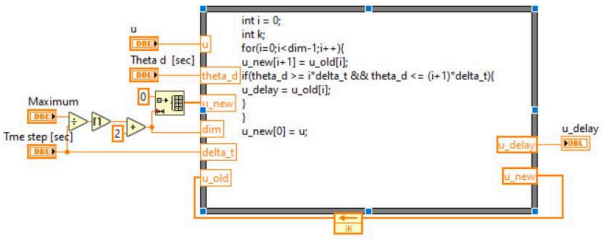


Figure 14: Sub VI for delay function

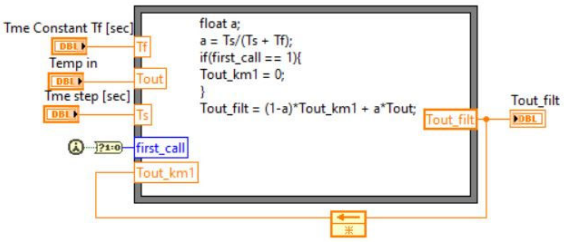


Figure 15: Sub VI for Low pass filter

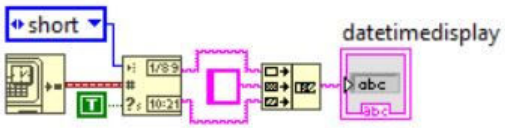


Figure 16: Sub VI for display date and time

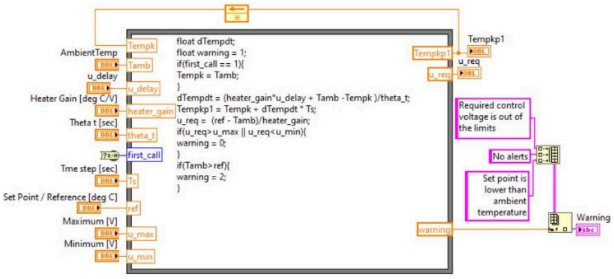


Figure 17: Sub VI for mathematical model simulator

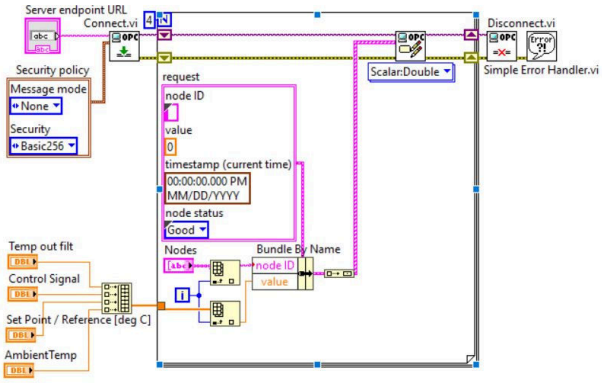


Figure 18: Sub VI for OPC writing

In this LabVIEW program, user can select mathematical model (simulator) or real air heater by switich in the lever. Mathematical model use the equation which is in the figure 6 and it has been included in the information page in the LabVIEW program. Formula node in the figure 17 use this equation and the differential equation is solved by using Euler Forward Method.

When using the real air heater, scaling is required when converting voltage output into temperature. In figure 10, analog output is a voltage output in 1-5V range. This should be converted into temperature value. Equivalent temeprature range for the voltage output is 0-50 deg C. By considering the linear relationship between voltage and the temeprature, folloing equation can be generated.

Temperature = 12.5*Voltage – 12.5

This can be defined in the DAQ assistant in the LabVIEW program as in the figure 19. No scaling is required for the control signal, since it is a voltage signal. Wiring connection was carried out as in the figure 20.

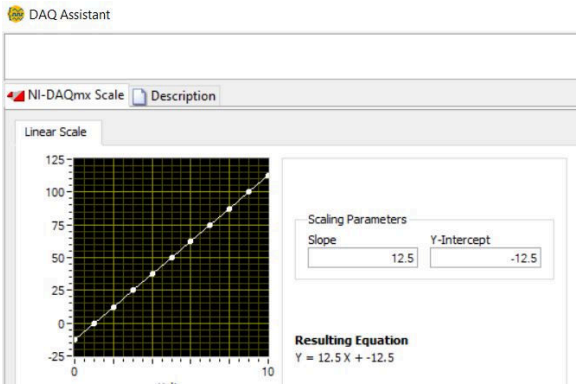


Figure 19: Set parameters for linear relationship between output voltage and temperature

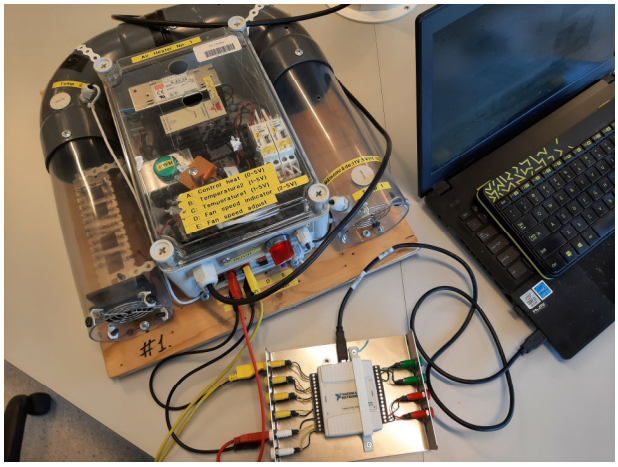


Figure 20: Wiring connection between air heater, USB 6008 and PC

Main object of control system is its PID controller. For the PID controller, following equation can be defined as the relationship between control error and the control voltage by using descritization (See figure 21). This equation has been used in the formula node in the figure 13. Since control voltage should lie between 0-5V range, additonal statement has been included in the formula node to limit this voltage range.

- Manual control signal:
$$u_{\mathrm{man},k} = \text{constant}$$
- P term:
$$u_{p,k} = K_c e_k$$
- I term:
$$u_{i,k} = u_{i,k-1} + \frac{K_c T_s}{T_i} e_k$$
- D term:
$$u_{d,k} = K_c T_d \frac{e_k - e_{k-1}}{T_s}$$
- Total control signal:
$$u_k = u_{\mathrm{man}} + u_{p,k} + u_{i,k} + u_{d,k}$$

Figure 21: PID controller equations [2]

Low pass filter is used eliminate the noise in the measured signal. In this study also, a low pass filter has been used to overcome measuring noises. Equation for the low pass filter has been illustrated in figure 22. Sub VI shown in the figure 15 was used for the low pass filter in this control application.

$$\frac{T_s}{T_f + T_s} \equiv a$$

$$y_k = (1 - a)y_{k-1} + au_k$$

Filter output

Noisy input signal

$$T_s \leq \frac{T_f}{5}$$

Figure 22: Low pass filter equation [1]

Delay function has been created in a simple way (Figure 14). LabVIEW program store the history values of the control voltage by an array and when new control voltage generated, oldest value is deleted from the array and all values are shifted by one to higher position index in the array. New value is stored in the 0 index in the array. Figure 23 shows a graphical illustration of the delay in descrete form.

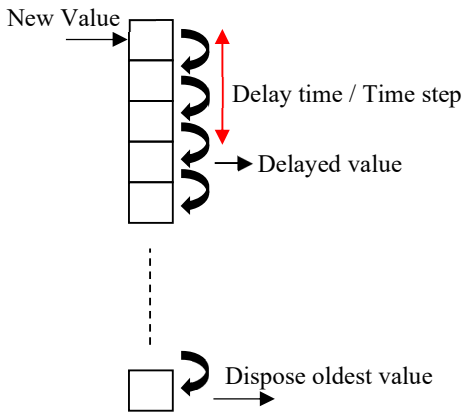


Figure 23: Illustration of delay function in discrete form

According to the figure 24, there is a maximum values that can be stored since no of elements in the array is fixed. Therefore, maximum delay value has been intorduced in the application and highlighted to see easily for the user (Figure 5).

While controlling the temeprature of air heater, this application parallely work as a OPC client. It connect with a OPC server as describes in section IV. The application pass four different values to the OPC server such as Output temperature, Control Voltage, Ambient Temperature and Set point. For each value, there is a separate node created by the OPC server and thouse nodes are used for the communication. OPC communication starts automatically when user started the controlling process. Refer figures 11 and 18 to find out the block diagrams for OPC communication.

Figure 24 shows controlling the temeprature when connected to the air heater.

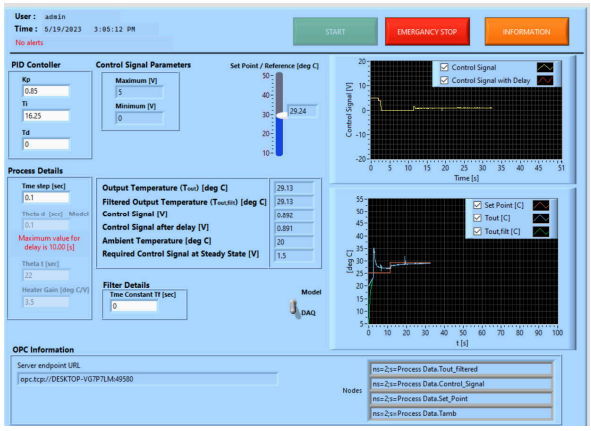


Figure 24: Controlling the air heater temperature with zero filter constant - After tuning PID controller

According to the figure 24, output temeprature reaches to the set point properly. However, since filter time constant is zero, it show some noises in the plot and this can be eliminated by adding a values according to the figure 22. Here PID controller is a PI controller. That means the derivative term is zero and no effect to the controlling part. When connected to the real air heater, no purpose of changing delay values and heater gains. Therefore, those controls become disable when switching from model to the DAQ and they become enable when switched into model simulator.

In the application, a value display with the tag “Required Control Signal at Steady State”. That value calculated by the mathematical model. At steady state, rate of change in output temeprature is zero. By considering this, it can be calculated the required control voltage by using the formula in figure 6. This value display as the control signal at steady state.

In the temeprature chart, user can compare the values such as output temperature, filtered temeprature and temeprature set point. Also, user can hide unwanted plots.

Information page of this application act as a catalog for the user. It shows how the application work with mathematical model and the real air heater. Also, it provides relavant equations such as mathematical model of the air heater and PID controller model which have been used in the application. Also, it has some notations which have been used in the controls in the controlling page. Therefore, user can get a clear picture before start the controlling part.

The application displays the logged user and the time at the top of the interface.

B. PID Controller Tuning

It is essential to carry out PID controller tuning before use in the real air heater. If not, control signal does not behave smoothly and there will be an error between reference point and the temperature output. There are different methods to tune the PID controller. In this method derivative term was ignored and PI controller was taken to tune. In this study, Relaxed Ziegler-Nichol's method was used to tune the PI controller. Ultimate gain and ultimate period were 3.4 and 13 seconds respectively. Equations in the following image were used to calculate K and T_i.

PI settings	K_c	T_i
Relaxed Ziegler-Nichols	$0.25K_{cu}$	$1.25P_u$
Original Ziegler-Nichols	$0.45K_{cu}$	$\frac{P_u}{1.2}$

Figure 25: Calculation of PI settings [2]

Calculate K and T_i was 0.85 and 16.25(See figure 5).

C. OPC Communication

To send values from control system to data logger, OPC communication has been used. Here, OPC UA type has been selected. Four types of values are sent through the OPC which are output temperature, control voltage, reference point and ambient temperature. There is a separate node for each value. Server address and nodes have been defined in the LabVIEW program permanently (See figure 5 OPC Information). Block diagram for the OPC communication can be seen in the figure 18.

D. Display warnings

Control system displays some warnings according to the user activities as shown in figure 26 and 27.

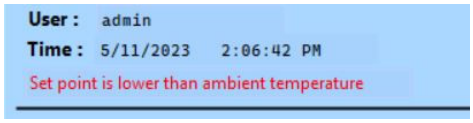


Figure 26 : Display warning for low set point

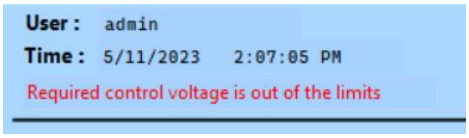


Figure 27: Display warning for high set point

IV. OPC UA Server

LabVIEW program was created as a OPC UA Server simulator with 4 nodes to pass the values from control system software to the data logger (See figure 28 and 29).



Figure 28: OPC UA server interface

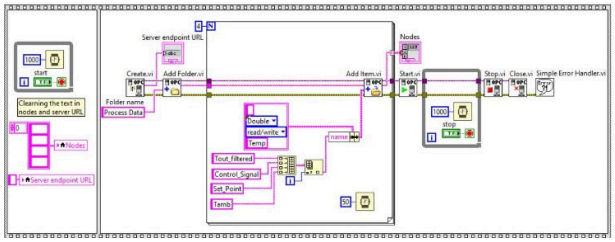


Figure 29: Block diagram of the OPC server

V. Designing and Implementation of Database

In this SCADA System, it requires a database with 4 tables as follows.

- To store login details
- To store process parameters
- To store alarms
- To store limits of the process parameters to generate alarms (High and Low values)

To design the database with relations, Erwin Data Modeler was used (See figure 30). Database was implemented in SQL server. In the database, login details are created by the system administrator. When login into the system, it is essential to retrieve a record which same as the given username and password. To perform this, stored procedure was created in the SQL database. Hence, when login by LabVIEW or C# application or ASP.NET application, this stored procedure executes and retrieve the corresponding record. Figure 31 shows the creation of stored procedure in SQL database. Process parameters are updated by the data logger. To update the alarm, a trigger was created. Purpose of this is, when process table is updating, trigger try to identify each process variable is within the limits which are stored in the limits table. When trigger detects any variable value out of the limits, a record creates in the alarm table. Process variables limits also can be updated by system administrator. Figure 32 shows a trigger created to set ambient temperature as the minimum set point limit. Figure 33 shows the created trigger to update the alarm table. This trigger creates an alarm record, only the last alarm in same types has been acknowledged.

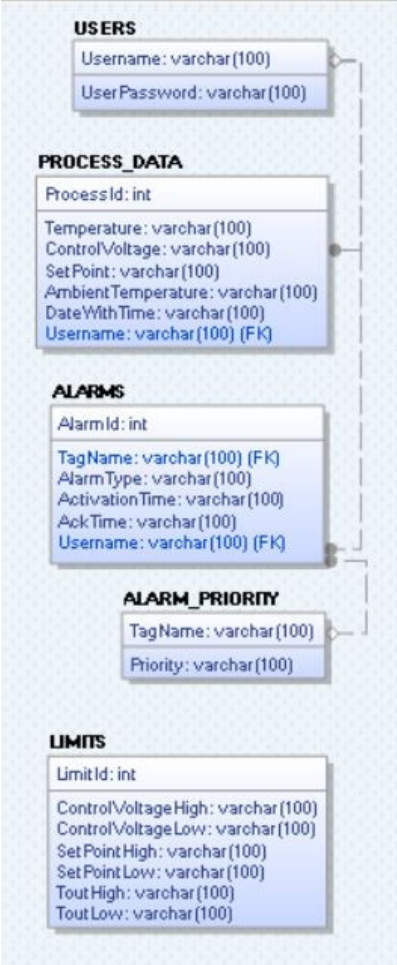


Figure 30: Database designing by Erwin Data Modeler

```
ALTER PROCEDURE [dbo].[UserDetails]
@Username varchar(50),
@UserPassword varchar(50)

AS
select * from USERS where
Username = @Username and
UserPassword = @UserPassword
```

Figure 31: Stored procedure to verify user at login

```
USE [AIR_HEATER]
GO
/****** Object:  Trigger [dbo].[GetSetPointLimits]    Script Date:
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[GetSetPointLimits] ON [dbo].[PROCESS_DATA]
FOR INSERT
AS
DECLARE
@AmbientTemp float

select @AmbientTemp = AmbientTemperature from INSERTED

update LIMITS set SetPointLow = @AmbientTemp
```

Figure 32: Trigger to set ambient temperature as the lowest temperature

```
AS
DECLARE
@ProcessId int,
@Temperature float,
@ControlVoltage float,
@SetPoint float,
@ToutHigh float,@ToutLow float,
@SetPointHigh float,
@SetPointLow float,
@ControlVoltageHigh float,
@ControlVoltageLow float,
@DateWithTime varchar(100),
@LastTempHighAck varchar(100),@LastTempLowAck varchar(100),@LastCVHighAck varchar(100),@LastCVLowAck varchar(100),
@LastSPHighAck varchar(100),@LastSPLowAck varchar(100),
@CountTempHigh int, @CountTempLow int, @CountCVHigh int, @CountCVLow int, @CountSPHigh int, @CountSPLow int

select @ProcessId = ProcessId from INSERTED
select @Temperature = Temperature, @ControlVoltage = ControlVoltage, @SetPoint = SetPoint,
@DateWithTime = DateWithTime from PROCESS_DATA where ProcessId = @ProcessId
select @ToutHigh = ToutHigh,@ToutLow = ToutLow, @SetPointHigh=SetPointHigh, @SetPointLow=SetPointLow,
@ControlVoltageHigh=ControlVoltageHigh, @ControlVoltageLow = ControlVoltageLow from LIMITS

select @LastTempHighAck = AckTime from ALARMS where AlarmType = 'High' and TagName = 'Output Temperature'
select @LastTempLowAck = AckTime from ALARMS where AlarmType = 'Low' and TagName = 'Output Temperature'
select @LastCVHighAck = AckTime from ALARMS where AlarmType = 'High' and TagName = 'Control Voltage'
select @LastCVLowAck = AckTime from ALARMS where AlarmType = 'Low' and TagName = 'Control Voltage'
select @LastSPHighAck = AckTime from ALARMS where AlarmType = 'High' and TagName = 'Set Point'
select @LastSPLowAck = AckTime from ALARMS where AlarmType = 'Low' and TagName = 'Set Point'
select @CountTempHigh = count(*) from ALARMS where AlarmType = 'High' and TagName = 'Output Temperature'
select @CountTempLow = count(*) from ALARMS where AlarmType = 'Low' and TagName = 'Output Temperature'
select @CountCVHigh = count(*) from ALARMS where AlarmType = 'High' and TagName = 'Control Voltage'
select @CountCVLow = count(*) from ALARMS where AlarmType = 'Low' and TagName = 'Control Voltage'
select @CountSPHigh = count(*) from ALARMS where AlarmType = 'High' and TagName = 'Set Point'
select @CountSPLow = count(*) from ALARMS where AlarmType = 'Low' and TagName = 'Set Point'

if(@Temperature > @ToutHigh and (@LastTempHighAck IS NOT NULL or @CountTempHigh = 0))
insert into ALARMS(TagName, AlarmType, ActivationTime) values('Output Temperature','High',@DatewithTime)

if(@Temperature < @ToutLow and (@LastTempLowAck IS NOT NULL  or @CountTempLow = 0))
insert into ALARMS(TagName, AlarmType,  ActivationTime) values('Output Temperature','Low',@DatewithTime)

if(@SetPoint > @SetPointHigh and (@LastSPHighAck IS NOT NULL  or @CountSPHigh = 0))
insert into ALARMS(TagName, AlarmType,  ActivationTime) values('Set Point','High',@DatewithTime)

if(@SetPoint < @SetPointLow and (@LastSPLowAck IS NOT NULL  or @CountSPLow = 0))
insert into ALARMS(TagName, AlarmType,  ActivationTime) values('Set Point','Low',@DatewithTime)

if(@ControlVoltage > @ControlVoltageHigh and (@LastCVHighAck IS NOT NULL  or @CountCVHigh = 0))
insert into ALARMS(TagName, AlarmType,  ActivationTime) values('Control Voltage','High',@DatewithTime)

if(@ControlVoltage < @ControlVoltageLow and (@LastCVLowAck IS NOT NULL  or @CountCVLow = 0))
insert into ALARMS(TagName, AlarmType, ActivationTime) values('Control Voltage','Low',@DatewithTime)
```

Figure 33: Trigger to update alarms

VI. Data Logger

Purpose of this to read data of the control system through the OPC server and insert into the relevant table. This consist of 4 pages which are system login page, data logging page, add new user page and view past records page. These pages are shows in figures 34, 35, 36 and 37 respectively.

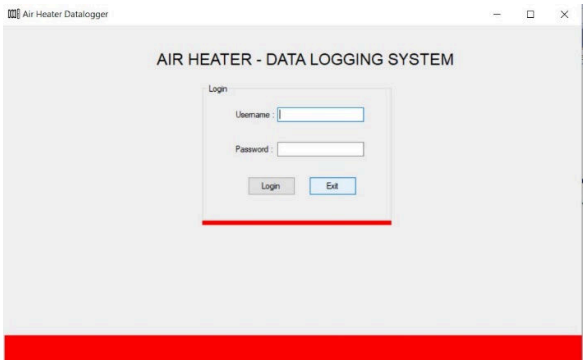


Figure 34: Login page of the data logger

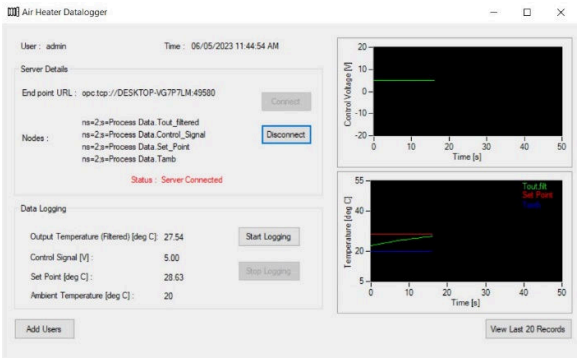


Figure 35: Data logging page

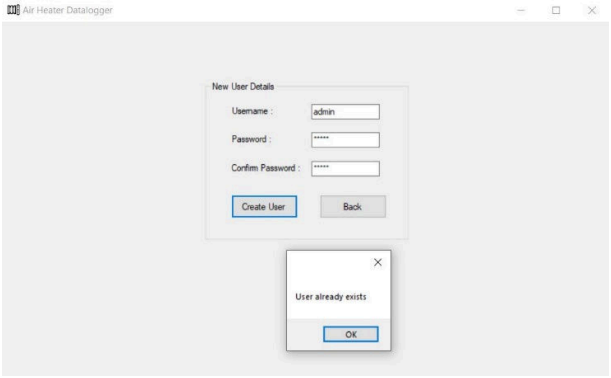


Figure 36: Add new user page and giving error message for already existing user creation

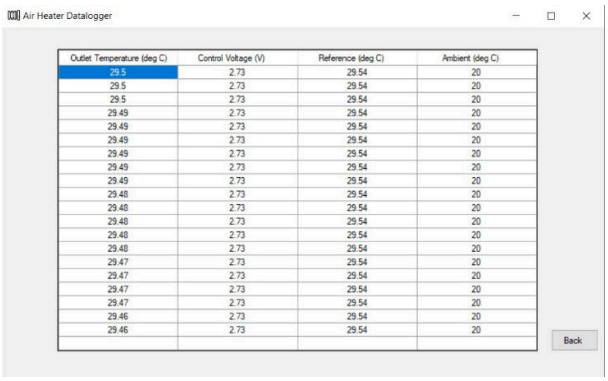


Figure 37: Displaying last 20 records

At the login, stored procedure executes. At the datalogging page, first need to connect with OPC server. Only the system administrator can add new users and these users can access control system, data logging system and alarming system. When logged into the system, use can see last 20 record by moving to another page. It was developed by C#.

In this data logging application, which was developed by using C# has additional class to communicate with the SQL server. As in the figure 35, application shows the username who logged into the system. Also, when user started data logging, all new records include the username to identify the which user involved for the datalogging. Datalogging time also records in the database. As mentioned above, only the system administrator can add new users. For all other users, "Add Users" button not visible (See figure 38). When developing the software, tab control was used to move between pages rather than using new window for each page. This has increased the consistency of software and it will not disturb the eye while working in the software. If user disconnected with communication with OPC server, data logging also stops automatically without making any errors.

VII. ALARM SYSTEM

This is a web application and to develop this, ASP.NET Core MVC (Model – View - Controller) frame work was used.

Figure 39, 40, 41 and 42 shows the interface of the application. Communication between SQL server and application has been done through AJAX request (Figure 43).

```
private void btnLogin_Click(object sender, EventArgs e)
{
    if (SQL_Reader_Writer.read_login(txtUsername.Text, txtPassword.Text))
    {
        MessageBox.Show("Login successful", "Success", MessageBoxButtons.OK);
        lblUser.Text = txtUsername.Text;
        tabControlMain.SelectedIndex = 1;
        timerDateTime.Start();
        if (txtUsername.Text == "admin") {
            btnAddUsers.Show();// Show add users only for admin login
        }
    }
```

Figure 38: Give permission to add users to administrator only

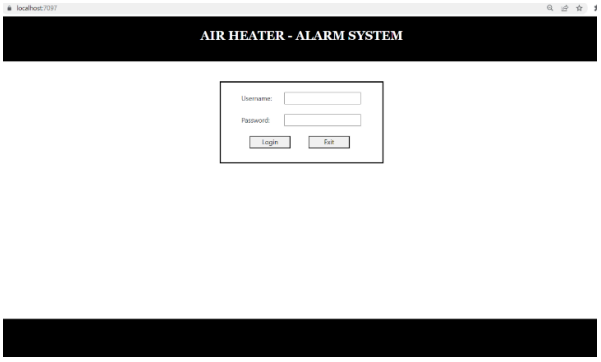


Figure 39:Login page of the alarm system

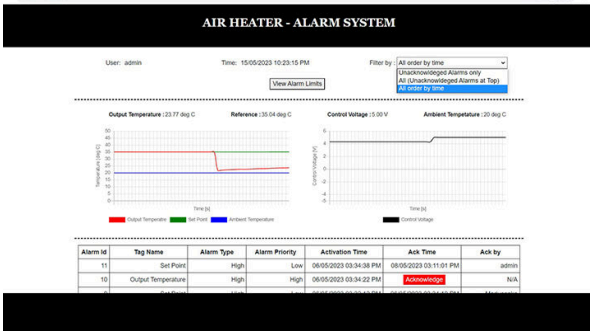


Figure 40:Showing the data in graphs for the user

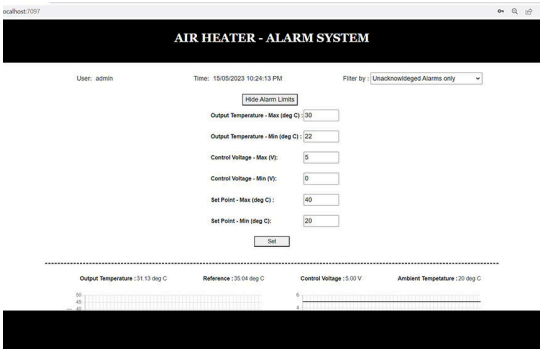
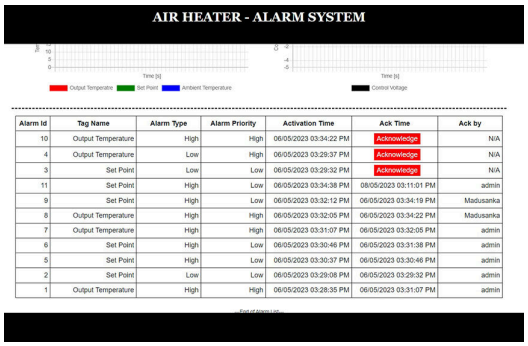


Figure 41: Viewing alarm limits



Alarm Id	Tag Name	Alarm Type	Alarm Priority	Activation Time	Ack Time	Ack by
10	Output Temperature	High	High	06/05/2023 03:34:22 PM	Acknowledge	N/A
4	Output Temperature	Low	High	06/05/2023 03:29:37 PM	Acknowledge	N/A
3	Set Point	Low	Low	06/05/2023 03:29:32 PM	Acknowledge	N/A
11	Set Point	High	Low	06/05/2023 03:34:38 PM	08/05/2023 03:11:01 PM	admin
9	Set Point	High	Low	06/05/2023 03:32:12 PM	06/05/2023 03:34:19 PM	Madusanka
8	Output Temperature	High	High	06/05/2023 03:32:05 PM	06/05/2023 03:34:22 PM	Madusanka
7	Output Temperature	High	High	06/05/2023 03:31:07 PM	06/05/2023 03:32:05 PM	admin
6	Set Point	High	Low	06/05/2023 03:30:46 PM	06/05/2023 03:31:38 PM	admin
5	Set Point	High	Low	06/05/2023 03:30:37 PM	06/05/2023 03:30:46 PM	admin
2	Set Point	Low	Low	06/05/2023 03:29:08 PM	06/05/2023 03:29:32 PM	admin
1	Output Temperature	High	High	06/05/2023 03:28:35 PM	06/05/2023 03:31:07 PM	admin

Figure 42: Showing the alarm list - Filtered according to the unacknowledged alarms at top

```
$.ajax({
    type: "POST",
    url: "/Home/Cont_Login",
    data: { //Passing data
        username: username, //Reading text box values using Jquery
        password: password
    },
    success: function (result) {
        if (result == "success") {
            alert("Login Successful");
            $("#login_section").hide();
            $("#logged_user").html(username);
            $("#datetime").html(get_date() + " " + get_time());
            get_alarms();
            $("#alarm_section").fadeIn(1000);;
            //$("#alarm_section").show();

        }
        else {
            alert(result);
        }
    },
```

Figure 43: Sending AJAX in the login state

This application also has a login page and the same users can log into this system as a control system and data logger. When the user logged into the system, all alarms can be filtered according to 3 types which are unacknowledged alarms only, all alarms (unacknowledged alarms at the top), and all alarms ordered by time. When a user clicked the red button for acknowledgment, it will update the database with the user and the acknowledged time (See figure 42). Also, users can view the alarm limits, but only the system administrator can change the alarm limits. For other users, the "Set" button is not visible (See figure 44).

```
$('#alarm_limits').text("Hide Alarm Limits");
if ($("#logged_user").html() == "admin") {
    $("#alarm_limits_set").css("visibility", "visible");
}
```

Figure 44: Giving permission to the administrator to change alarm limits

Purpose of showing graphs in the Alarm Application is (as in figure 40), when there is a new alarm, operator can easily identify whether this alarm continue or not. Also, he can take decisions easily.

VIII. Cyber Security

In this SCADA system, there are 3 applications which are control system, data logger and alarm system. All these applications consist of login requirement before starting the activities (See figures 4, 34, 39). Therefore, all applications come up with a security. Also, when performing a task, responsible user is also recorded by the application. Some activities can only be done by the system administrator only such as creating new users and changing the alarm limits. SQL database also consist of password to enhance the security of the database. Following also can be introduced to increase the security of the SCADA System.

- Changing password option for the users
- Inactive user deletion option for the adminstrator
- Lock the logging for each user after attempting 3 times with wrong password
- Encryption of data which is sent to the OPC server by control system and decryption by the data logger. If not, all OPC clients can access the data when server address and nodes are known.

IX. Conclusion

This SCADA system was developed to control the temperature of air heater. Process parameters are sent to the data logger via OPC server. Data logger store these process variables in the SQL database. The alarm system separately run and generate alarms with respect to the alarm limits and alarm priority. OPC server simulator is also created by the developer. SQL database has triggers and stored procedures to optimize the tasks. This system has been developed with some security such as login requirements and only administrator activities.

X. References

[1] H.-P. Halvorsen, "SCADA Systems - Assignment," Porsgrunn, 2023.
[2] F. A. Haugen, Modeling, Simulation and Control, Porsgruun, 2022