

DIRECT CONTROL OF PROCESS EQUIPMENTS IN A Distributed Control System

By

Jobair Hasan Subbir and

Noushin Tabassum



Submitted to the

Department of Electrical and Electronic Engineering
Faculty of Sciences and Engineering
East West University

In partial fulfillment of the requirements for the degree of
Bachelor of Science in Electrical and Electronic Engineering
(B.Sc. in EEE)

Fall, 2010

Approved By

A handwritten signature in blue ink, appearing to be "KMR", written over a horizontal line.

Thesis Advisor
Dr. Kazi Mujibur Rahman

A handwritten signature in blue ink, appearing to be "AA Haque", written over a horizontal line.

Chairperson
Dr. Anisul Haque



Abstract

Distributed control systems (DCS) are employed in complex process control systems for monitoring, supervision, setpoint control and direct equipment control. In this project we concentrate on direct equipment control using a general purpose data acquisition module.

A software program with a graphical user interface (GUI) is designed using Visual Basic in .Net environment that involve three processes. Monitoring textboxes, setpoint spin controls and checkbox/button based direct equipment control is accomplished in the GUI design.

Control signals from the GUI program are sent to the process equipments using an Advantech USB-4711A DAQ module. Signals from the DAQ module are optically isolated and ON/OFF signals are passed to the process equipments through relay interface. The proposed technique of using general purpose DAQ hardware as a control device is supposed to be new in the era of DCS. The system is tested in the laboratory and is found to work satisfactorily.



Acknowledgements

In our project work we are grateful to those who contributed us with their valuable times and efforts.

At the very beginning, we will mention the name of our project supervisor Dr. Kazi Mujibur Rahman . We feel lucky and honored to get the chance of completing our project under his supervision. We learnt a lot from him regarding program development using visual studio and especially controlling hardware in real time using a computer. He provided us the information regarding different aspects of our project. We are also grateful to him also for reading, checking and reviewing our project report.

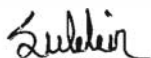
Special thanks to Dr. Anisul Haque, Chairperson, EEE, East West University for his guidelines, support and care.

We wish to express our appreciation to our parents, friends, teachers and lab-officers for their support.

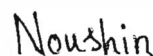
Suggestions for improvements in the project and the report are always welcome.

Authorization page

I hereby declare that I am the sole author of this thesis. I authorize East West University to lend this thesis to other institutions or individuals for the purpose of scholarly research.



Jobair Hasan Subbir



Noushin Tabassum

I further authorize East West University to reproduce this thesis by photocopy or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.



Jobair Hasan Subbir



Noushin Tabassum

Table of Contents

Abstract	2
Acknowledgements	3
Authorization page	4
Table of Contents	5
List of Tables.....	7
List of Figures	8
Chapter 1 Introduction	9
1.1 Elements.....	9
1.2 Applications	10
1.3 History.....	11
1.3.1 The Network Centric Era of the 1980s	13
1.3.2 The Application Centric Era of the 1990s.....	14
1.4 Problem Statement	16
1.5 Objectives of the Project.....	17
Chapter 2 Direct Control of Process Equipments in a DCS.....	18
2.1 Direct Control Scheme.....	18
2.2 Graphic User Interface Development.....	20
2.2.1 Dyeing Process.....	20
2.2.2 Boiler.....	21
2.2.3 Office Floor.....	21
Chapter 3 Graphic User Interface for DCS	23
3.1 Interfacing the USB4711A.....	23

3.1.1 AdvDIO.WriteDoChannel Method.....	23
3.1.2 AdvDIO.WriteDoPorts Method.....	24
3.2 GUI Form.....	25
3.3 Real Time Operation.....	31
3.4 Direct Control of Process Equipments.....	32
Chapter 4 Software Codes.....	33
4.1 Button Clicked Events.....	33
4.2 Complete Program Codes.....	34
4.2.1 DCS Main Form Codes.....	34
4.2.2 Process 1 Form in a Separate Window.....	39
4.2.3 Process 2 Program in a Separate Window.....	41
4.2.4 Global Variable Declarations.....	43
Chapter 5 Results.....	44
5.1 DCS Software.....	44
5.2 DCS Interface Hardware.....	46
5.3 System Operation.....	46
Chapter 6 Conclusions.....	49
6.1 Conclusions.....	49
6.2 Future Works.....	49
Bibliography.....	50

List of Tables

Table 1: VB.Net codes and associated tasks performed by USB4711A.....25

Table 2 Process control assignment to digital output line of USB-4711A32



List of Figures

Figure 2.1. Functional block diagram of the direct control arrangement of process equipment using Advantech USB4711A data acquisition module	19
Figure 2.2. Direct controls interface to process equipment through optical isolation	19
Figure 2.3 Typical dyeing process used in the DCS	20
Figure 2.4 Boiler process added to the DCS	21
Figure 2.5 Office floor under the control of DCS	22
Figure 3.1 GUI window of the DCS showing three processes, necessary GUI controls and Event Log	27
Figure 3.2 DCS showing process 1 operation with all the valves closed and pumps ON.	28
Figure 3.3 DCS showing all three processes in operation.	29
Figure 3.4 DCS showing whole system operation with event logs for all processes.	30
Figure 5.1 All processes in the DCS shown in same window.	44
Figure 5.2 DCS operation showing event logs.....	45
Figure 5.3 DCS showing process 1 in a separate window linked to the main GUI.....	45
Figure 5.4 DCS showing process 2 opened in a separate window.....	46
Figure 5.5 Experimental setup of the DCS hardware for direct equipment control.	48
Figure 5.6 DCS running from a laptop and interfaced to relay board.....	48

Chapter 1 Introduction

A **distributed control system** (DCS) refers to a control system usually of a manufacturing system, process or any kind of dynamic system, in which the controller elements are not central in location (like the brain) but are distributed throughout the system with each component sub-system controlled by one or more controllers. The entire system of controllers is connected by networks for communication and monitoring.

DCS is a very broad term used in a variety of industries, to monitor and control distributed equipment. It has found applications in almost all process industries like,

- Electrical power grids and electrical generation plants
- Environmental control systems
- Water management systems
- Oil refining plants
- Chemical plants
- Pharmaceutical manufacturing
- Dry cargo and bulk oil carrier ships

1.1 Elements

A DCS typically uses custom designed processors as controllers and uses both proprietary interconnections and communications protocol for communication. Input and output modules

and sends information to output modules. The input modules receive information from input instruments in the process (e.g., field) and transmit instructions to the output instruments in the field. Computer buses or electrical buses connect the processor and modules through multiplexer or de-multiplexers. Buses also connect the distributed controllers with the central controller and finally to the Human-Machine Interface (HMI) or control consoles.

Elements of a distributed control system may directly connect to physical equipment such as switches, pumps and valves or may work through an intermediate system such as a SCADA system.

1.2 Applications

Distributed Control Systems (DCSs) are dedicated systems used to control manufacturing processes that are continuous or batch-oriented, such as oil refining, petrochemicals, central station power generation, pharmaceuticals, food & beverage manufacturing, cement production, steelmaking, and papermaking. DCSs are connected to sensors and actuators and use setpoint control to control the flow of material through the plant. The most common example is a setpoint control loop consisting of a pressure sensor, controller, and control valve. Pressure or flow measurements are transmitted to the controller, usually through the aid of a signal conditioning Input/Output (I/O) device. When the measured variable reaches a certain point, the controller instructs a valve or actuation device to open or close until the fluidic flow process reaches the desired setpoint. Large oil refineries have many thousands of I/O points and employ very large DCSs. Processes are not limited to fluidic flow through pipes, however, and can also include things like paper machines and their associated quality controls, variable speed drives and motor control centers, cement kilns, mining operations, ore processing facilities, and many others.

A typical DCS consists of functionally and/or geographically distributed digital controllers capable of executing from 1 to 256 or more regulatory control loops in one control box. The input/output devices (I/O) can be integral with the controller or located remotely via a field network. Today's controllers have extensive computational capabilities and, in addition to proportional, integral, and derivative (PID) control, can generally perform logic and sequential control.

DCSs may employ one or several workstations and can be configured at the workstation or by an off-line personal computer. Local communication is handled by a control network with transmission over twisted pair, coaxial, or fiber optic cable. A server and/or applications processor may be included in the system for extra computational, data collection, and reporting capability.

1.3 History

Early minicomputers were used in the control of industrial processes since the beginning of the 1960s. The IBM 1800, for example, was an early computer that had input/output hardware to gather process signals in a plant for conversion from field contact levels (for digital points) and analog signals to the digital domain.

The first industrial control computer system was built 1959 at the Texaco Port Arthur, Texas, refinery with an RW-300 of the Ramo-Wooldridge Company [1].

The DCS was introduced in 1975. Both Honeywell and Japanese electrical engineering firm Yokogawa introduced their own independently produced DCSs at roughly the same time, with the TDC 2000 and CENTUM [2] systems, respectively. US-based Bristol also introduced their UCS 3000 universal controller in 1975. In 1980, Bailey (now part of ABB [3]) introduced the NETWORK 90 system. Also in 1980, Fischer & Porter Company (now

also part of ABB [4]) introduced DCI-4000 (DCI stands for Distributed Control Instrumentation).

The DCS largely came about due to the increased availability of microcomputers and the proliferation of microprocessors in the world of process control. Computers had already been applied to process automation for some time in the form of both Direct Digital Control (DDC) and Set Point Control. In the early 1970s Taylor Instrument Company, (now part of ABB) developed the 1010 system, Foxboro the FOX1 system and Bailey Controls the 1055 systems. All of these were DDC applications implemented within minicomputers (DEC PDP-11, Varian Data Machines, MODCOMP etc.) and connected to proprietary Input/Output hardware. Sophisticated (for the time) continuous as well as batch control was implemented in this way. A more conservative approach was Set Point Control, where process computers supervised clusters of analog process controllers. A CRT-based workstation provided visibility into the process using text and crude character graphics. Availability of a fully functional graphical user interface was a way away.

Central to the DCS model was the inclusion of control function blocks. Function blocks evolved from early, more primitive DDC concepts of "Table Driven" software. One of the first embodiments of object-oriented software, function blocks were self contained "blocks" of code that emulated analog hardware control components and performed tasks that were essential to process control, such as execution of PID algorithms. Function blocks continue to endure as the predominant method of control for DCS suppliers, and are supported by key technologies such as Foundation Fieldbus [5] today.

Digital communication between distributed controllers, workstations and other computing elements (peer to peer access) was one of the primary advantages of the DCS. Attention was duly focused on the networks, which provided the all-important lines of communication that, for process applications, had to incorporate specific functions such as determinism and

redundancy. As a result, many suppliers embraced the IEEE 802.4 networking standard. This decision set the stage for the wave of migrations necessary when information technology moved into process automation and IEEE 802.3 rather than IEEE 802.4 prevailed as the control LAN.

1.3.1 The Network Centric Era of the 1980s

The DCS brought distributed intelligence to the plant and established the presence of computers and microprocessors in process control, but it still did not provide the reach and openness necessary to unify plant resource requirements. In many cases, the DCS was merely a digital replacement of the same functionality provided by analog controllers and a panel board display. This was embodied in The Purdue Reference Model (PRM) that was developed to define Manufacturing Operations Management relationships. PRM later formed the basis for ISA95 standards activities today.

In the 1980s, users began to look at DCSs as more than just basic process control. A very early example of a Direct Digital Control DCS was completed by the Australian business Midac in 1981-1982 using R-Tec Australian designed hardware. The system installed at the University of Melbourne used a serial communications network, connecting campus buildings back to a control room "front end". Each remote unit ran 2 Z80 microprocessors whilst the front end ran 11 in a Parallel Processing configuration with paged common memory to share tasks and could run up to 20,000 concurrent controls objects.

It was believed that if openness could be achieved and greater amounts of data could be shared throughout the enterprise that even greater things could be achieved. The first attempts to increase the openness of DCSs resulted in the adoption of the predominant operating system of the day: *UNIX*. UNIX and its companion networking technology TCP-IP were developed by the Department of Defense for openness, which was precisely the issue the process industries were looking to resolve.

As a result suppliers also began to adopt Ethernet-based networks with their own proprietary protocol layers. The full TCP/IP standard was not implemented, but the use of Ethernet made it possible to implement the first instances of object management and global data access technology. The 1980s also witnessed the first PLCs integrated into the DCS infrastructure. Plant-wide historians also emerged to capitalize on the extended reach of automation systems. The first DCS supplier to adopt UNIX and Ethernet networking technologies was Foxboro, who introduced the I/A Series system in 1987.

1.3.2 The Application Centric Era of the 1990s

The drive toward openness in the 1980s gained momentum through the 1990s with the increased adoption of Commercial off-the-shelf (COTS) components and IT standards. Probably the biggest transition undertaken during this time was the move from the UNIX operating system to the Windows environment. While the realm of the real time operating system (RTOS) for control applications remains dominated by real time commercial variants of UNIX or proprietary operating systems, everything above real-time control has made the transition to Windows.

The introduction of Microsoft at the desktop and server layers resulted in the development of technologies such as OLE for Process Control (OPC), which is now a de facto industry connectivity standard. Internet technology also began to make its mark in automation and the DCS world, with most DCS HMI supporting Internet connectivity. The '90s were also known for the "Fieldbus Wars", where rival organizations competed to define what would become the IEC fieldbus standard for digital communication with field instrumentation instead of 4-20 milliamp analog communications. The first fieldbus installations occurred in the 1990s. Towards the end of the decade, the technology began to develop significant momentum, with the market consolidated around Foundation Fieldbus and Profibus PA for process automation applications. Some suppliers built new systems from the ground up to maximize functionality

with fieldbus, such as Honeywell with Experion & Plantscape SCADA systems, ABB with System 800xA [6], Emerson Process Management [7] with the DeltaV control system, Siemens [8] with the Simatic PCS7 [9] and azbil [10] from Yamatake with the Harmonas-DEO system.

The impact of COTS, however, was most pronounced at the hardware layer. For years, the primary business of DCS suppliers had been the supply of large amounts of hardware, particularly I/O and controllers. The initial proliferation of DCSs required the installation of prodigious amounts of this hardware, most of it manufactured from the bottom up by DCS suppliers. Standard computer components from manufacturers such as Intel and Motorola, however, made it cost prohibitive for DCS suppliers to continue making their own components, workstations, and networking hardware.

As the suppliers made the transition to COTS components, they also discovered that the hardware market was shrinking fast. COTS not only resulted in lower manufacturing costs for the supplier, but also steadily decreasing prices for the end users, who were also becoming increasingly vocal over what they perceived to be unduly high hardware costs. Some suppliers that were previously stronger in the PLC business, such as Rockwell Automation and Siemens, were able to leverage their expertise in manufacturing control hardware to enter the DCS marketplace with cost effective offerings, while the stability/scalability/reliability and functionality of these emerging systems are still improving. The traditional DCS suppliers introduced new generation DCS System based on the latest Communication and IEC Standards, which resulting in a trend of combining the traditional concepts/functionalities for PLC and DCS into a one for all solution—named "Process Automation System". The gaps among the various systems remain at the areas such as: the database integrity, pre-engineering functionality, system maturity, communication transparency and reliability. While it is expected the cost ratio is relatively the same (the

more powerful the systems are, the more expensive they will be), the reality of the automation business is often operating strategically case by case. The current next evolution step is called Collaborative Process Automation Systems.

To compound the issue, suppliers were also realizing that the hardware market was becoming saturated. The lifecycle of hardware components such as I/O and wiring is also typically in the range of 15 to over 20 years, making for a challenging replacement market. Many of the older systems that were installed in the 1970s and 1980s are still in use today, and there is a considerable installed base of systems in the market that are approaching the end of their useful life. Developed industrial economies in North America, Europe, and Japan already had many thousands of DCSs installed, and with few if any new plants being built, the market for new hardware was shifting rapidly to smaller, albeit faster growing regions such as China, Latin America, and Eastern Europe.

Because of the shrinking hardware business, suppliers began to make the challenging transition from a hardware-based business model to one based on software and value-added services. It is a transition that is still being made today. The applications portfolio offered by suppliers expanded considerably in the '90s to include areas such as production management, model-based control, real-time optimization, Plant Asset Management (PAM), Real Time Performance Management (RPM) tools, alarm management, and many others. To obtain the true value from these applications, however, often requires a considerable service content, which the suppliers also provide.

1.4 Problem Statement

DCS is a huge and complex system where manufacturers use their own dedicated hardware and communication protocols. The tasks undertaken by a DCS can be summarized as follows:

- i) Process monitoring

- ii) Setting of process setpoints
- iii) Connect to physical equipment such as switches, pumps and valves or may work through an intermediate system such as a SCADA system.
- iv) Directly connect to physical equipment such as switches, pumps and valves.

An important part of the DCS is direct connection to physical equipments of processes. The direct control functionality can be programmed and tested in lab without the need for expensive and sophisticated DCS hardware.

Our focus in this project is to work on the direct physical connection and control part of a DCS.

1.5 Objectives of the Project

The objectives of this project are to

- a) Get familiarization with the literature of DCS,
- b) Develop the functional part of a DCS using general purpose hardware,
- c) Implement the direct control function in small scale using an industrial grade data acquisition module,
- d) Design and test the related hardware interface and develop GUI interface using modern .Net based visual programming.





Chapter 2 Direct Control of Process Equipments in a DCS

Custom built DCS uses specialized hardware for process access, communication and direct control purposes. In this project we intend to use a general purpose data acquisition module that would undertake the direct control part. We consider that other monitoring information from different processes is available as variables supplied through Activex or Dynamic Linking Library.

2.1 Direct Control Scheme

Figure 2.1 shows the direct control scheme developed using Advantech USB 4711A data acquisition (DAQ) module. The DAQ is connected to the laptop PC through USB. Process equipments direct control is accomplished by the digital output lines. USB 4711A have 8 output and 8 input lines. Upto 8 equipment can be controlled by a single DAQ USB 4711A. However for more controls, several USB-4711A can be connected to the PC. In case of limited USB ports in a Laptop/PC, USB hub can be used to add more DAQ modules.

The I/O control lines connect the process equipments through optically isolated drives as shown in Figure 2.2. The DAQ and the PC have common grounds but the process grounds are isolated using the 4N25 optical isolator. Relay outputs are used to ON/OFF the process equipments.

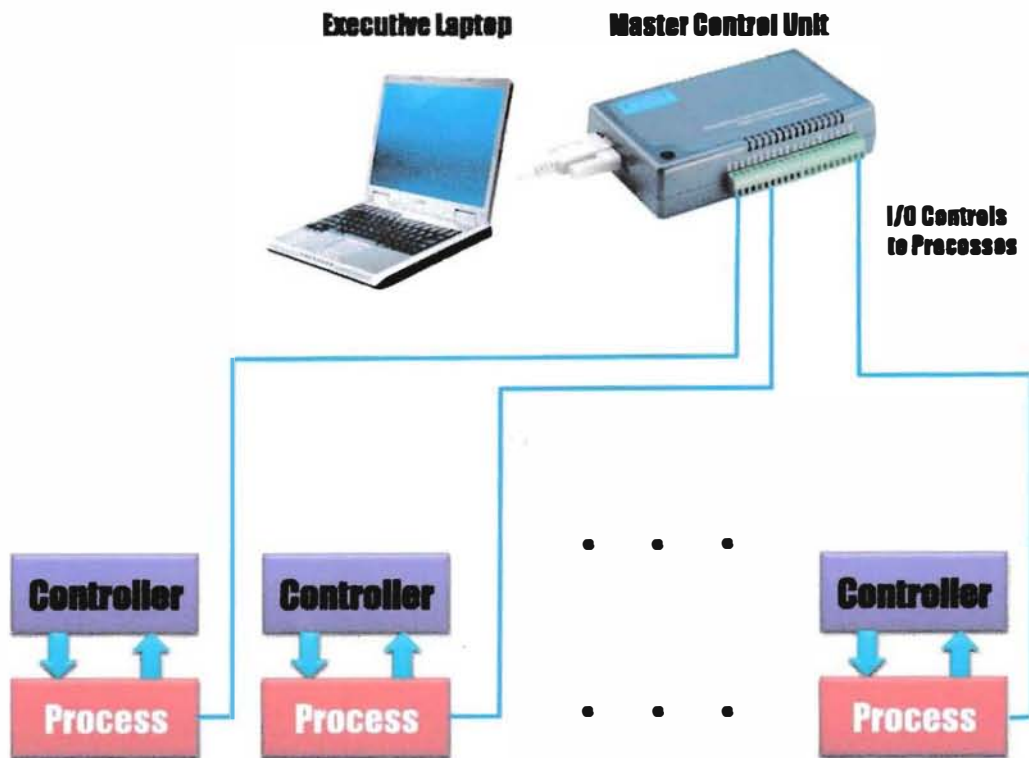


Figure 2.1. Functional block diagram of the direct control arrangement of process equipment using Advantech USB4711A data acquisition module

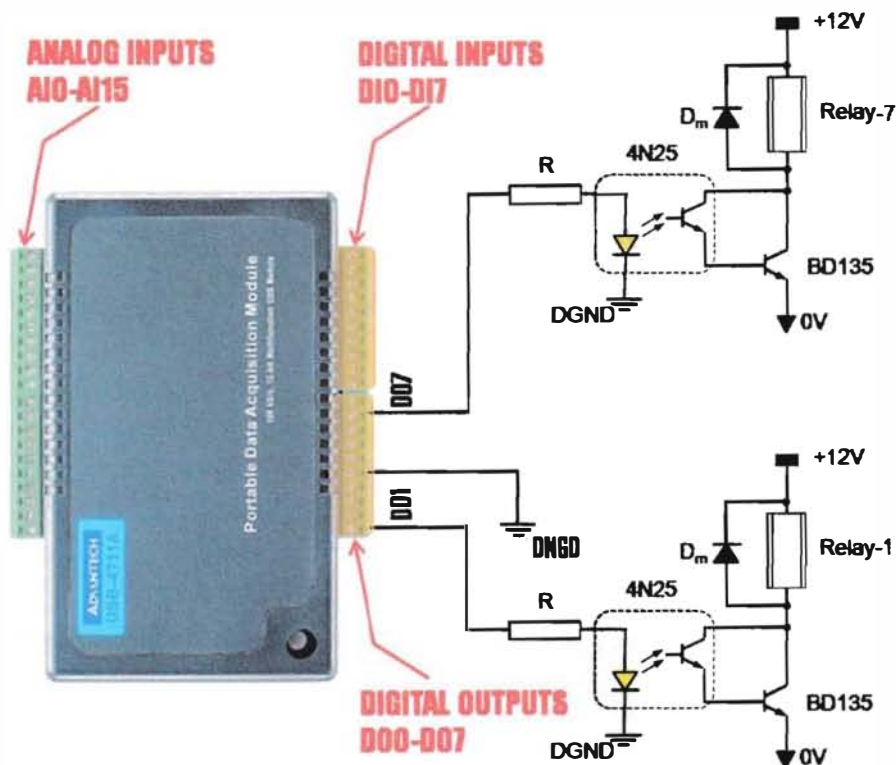


Figure 2.2. Direct controls interface to process equipment through optical isolation

The network based hardware is omitted in this project as our work is concentrated on the direct equipment control only.

2.2 Graphic User Interface Development

We considered two typical processes, one being a Dyeing process and the other one a boiler. The third one is not a process but a large office floor having spilt air conditioners that are also controlled from the DCS GUI. Graphic picture of the processes are embedded on the background of the main GUI form. Control buttons, check boxes, spin control and text boxes are inserted in appropriate places to demonstrate system monitoring and control clearly.

2.2.1 Dyeing Process

The dyeing process picture is shown in Figure 2.3. It contains dyeing autoclave, separator, work tank, pressurization pump, circulating pump, heat exchanger and filter. In addition there are a number of valves.

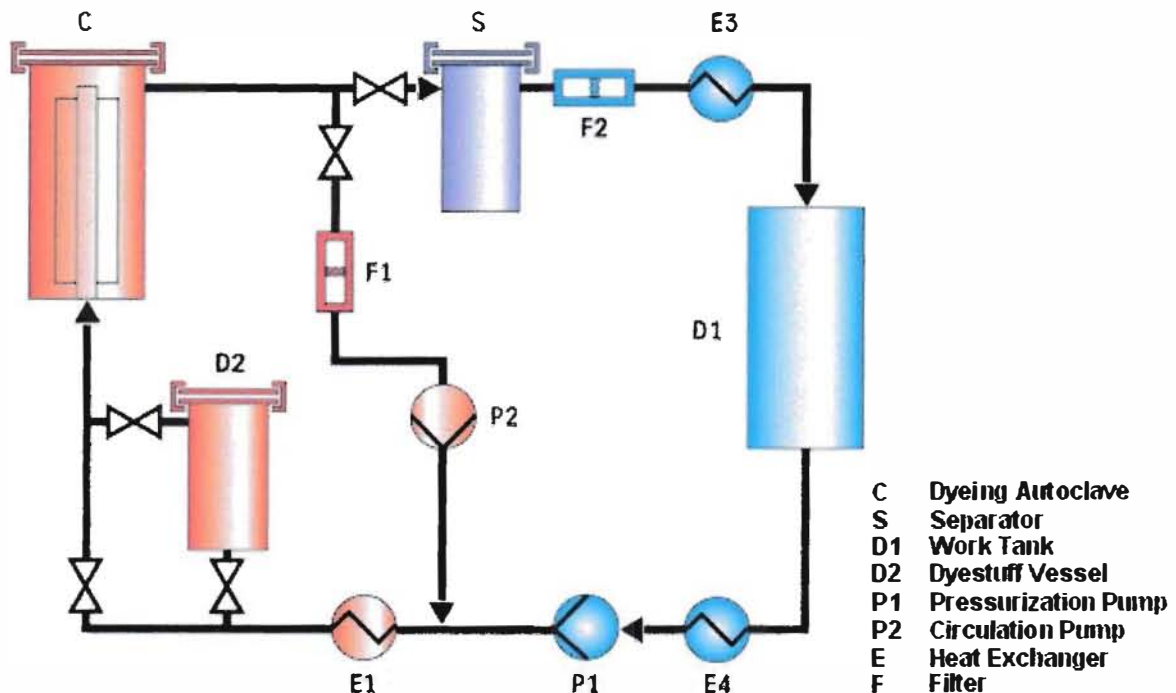


Figure 2.3 Typical dyeing process used in the DCS

The process invariably has monitoring parameters like temperatures in the heat exchanger.

We added direct controls to all the valves and start/stop of the pump motors.

2.2.2 Boiler

The second process is a boiler as shown in Figure 2.4. The boiler has feed water and makeup water intake pumps, water level control, steam pressure control and associated monitoring as well as the fuel supply controls. In our DCS we included monitoring of the steam temperature and pressure, water level setting, water level and two direct controls of the water pumps.

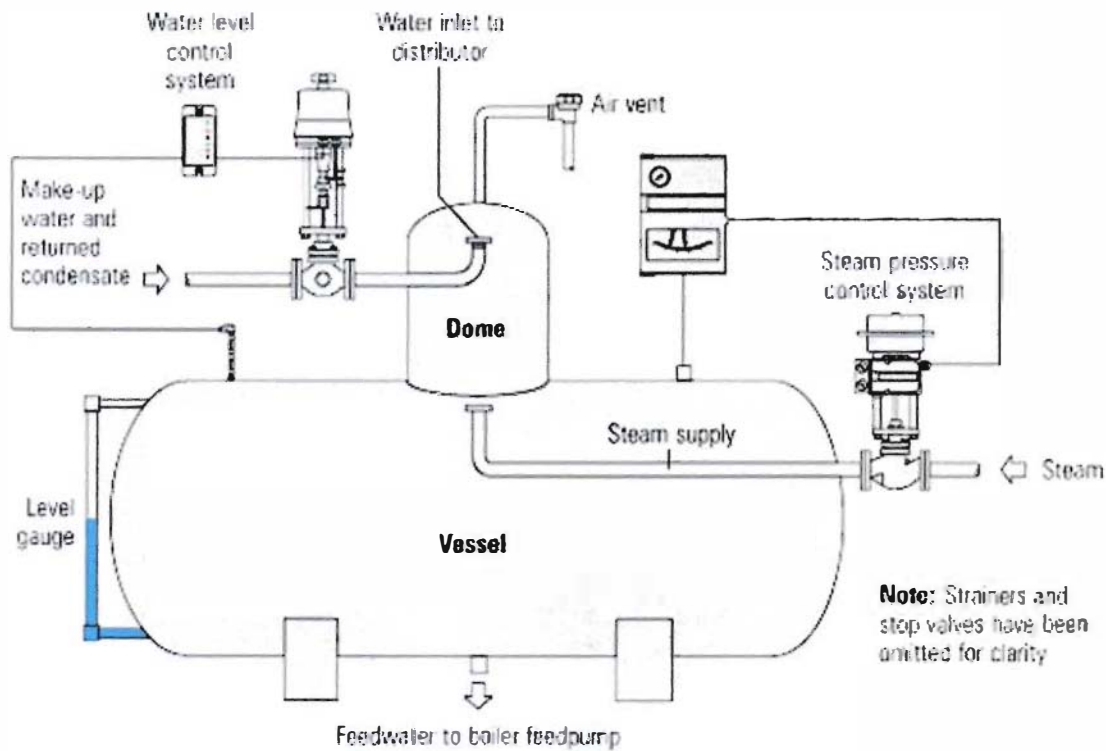


Figure 2.4 Boiler process added to the DCS

2.2.3 Office Floor

The third process we used is an office floor (Figure 2.5) that contains four AC's. These AC's are also considered under the direct control.

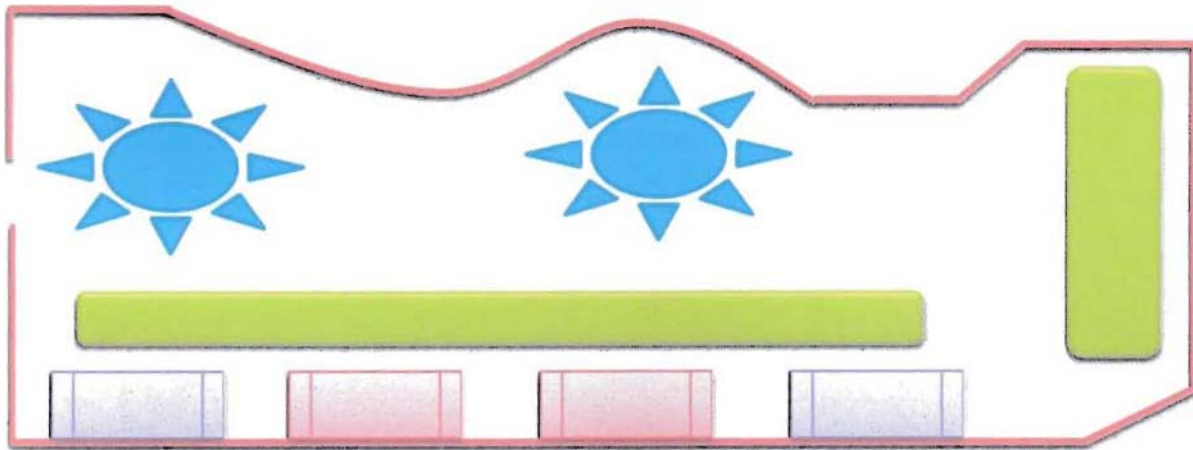


Figure 2.5 Office floor under the control of DCS

Once the processes direct controls from DCS and monitoring parameters are finalized the next part is the GUI based program development that would interactively communicate in hardware level without halting process operation.





Chapter 3 Graphic User Interface for DCS

The graphic user interface (GUI) is a windows form that hosts all the process outline pictures, data displays and controls accessible by mouse and keyboard. The procedures and key software codes used in the GUI are illustrated in this chapter.

3.1 Interfacing the USB4711A

The USB 4711A DAQ is the crucial interface used in this work that is primarily responsible for all direct controls. Since we are developing our GUI in .Net environment we utilized the .Net driver "AxInterop.AdvDIOLib.dll" supplied by Advantech. This driver communicates to the DAQ module through the USB port. Advantech supplies this driver through the ActiveDAQ Pro software utility.

The "AdvDIOLib" DLL supports two types of digital data outputs, (i) accessing channels independently using "WriteDoChannel" and (ii) accessing few channels together using "WriteDoPorts" methods.

3.1.1 AdvDIO.WriteDoChannel Method

This method outputs specific digital value on a specified DO channel.

Syntax

```
BOOL WriteDoChannel (  
    long status
```

```
    long channel  
)
```

Parameters

status, *IN* (The output digital value.)

channel, *IN* (The specified output DO channel.)

Return Value

True: if successful.

False: if failed.

3.1.2 AdvDIO.WriteDoPorts Method

This method outputs the data of specified DO ports.

Syntax

```
BOOL WriteDoPorts(  
    VARIANT* data,  
    long portStart,  
    long portCount = 1  
)
```

Parameters

data, *IN* (The buffer that stores the user specified data.)

portStart, *IN* (The starting DO port number. It is used to specify the DO port range.)

portCount, *IN* (The DO channel count. It is used to specify the DO port range.)

Return Value

True: if successful.

False: if failed.

The DLL file has to be included in the VB.Net program using normal procedures. We rename the device as "AxAdvDIO1" in our program. Table 1 shows a number of instructions supported by USB4711A in VB.Net.

Table 1: VB.Net codes and associated tasks performed by USB4711A

VB Code	Work Performed
AxAdvDIO1.WriteDoChannel(1, 0)	Writes 1 to DO0 channel
AxAdvDIO1.WriteDoChannel(0, 1)	Writes 0 to DO0 channel
AdvDIO1.WriteDoPorts(userData, 0, 8)	Writes userData to port from channel 0, total channels 8 (DO0-DO7)
AdvDIO1.WriteDoPorts(Data1, 0, 5)	Writes Data1 to port from channel 0, total channels 5 (DO0-DO4) other channels remain unaffected.

In the “WriteDoChannel” function, the first argument is the status and the second argument represents the channel. Thus “WriteDoChannel (1, 0)” writes 1 to channel DO0. On the other hand “WriteDoPorts” writes parallel data to specified channels. For example “WriteDoPorts(71,0,7)” writes to DO port bits D0 to D6, the bit D7 retains the old state.

3.2 GUI Form

A view of GUI form developed in this project is shown in Figure 3.1. For simplicity the three processes are outlined in the same window. An event log text box is placed in the form that displays all events (operation of any GUI control of any process with time stamp). The event log helps system administrator to keep track of the centralized command controls. The following GUI controls are added in the processes:

Process 1 (Dyeing Process)

- ❖ Five checkboxes for valves (assuming the valves are solenoid controlled),
- ❖ Three textboxes for displaying process temperatures,
- ❖ Two buttons for pump motors.

Process 2 (Boiler)

- ❖ One spin box for setting water level,
- ❖ Two textboxes for temperatures with one textbox showing pressure as well,
- ❖ Two buttons, where one button starts the boiler and the other operates the boiler feed pump.

Process 3 (Office Floor)

- ❖ Four buttons for operating split air coolers.

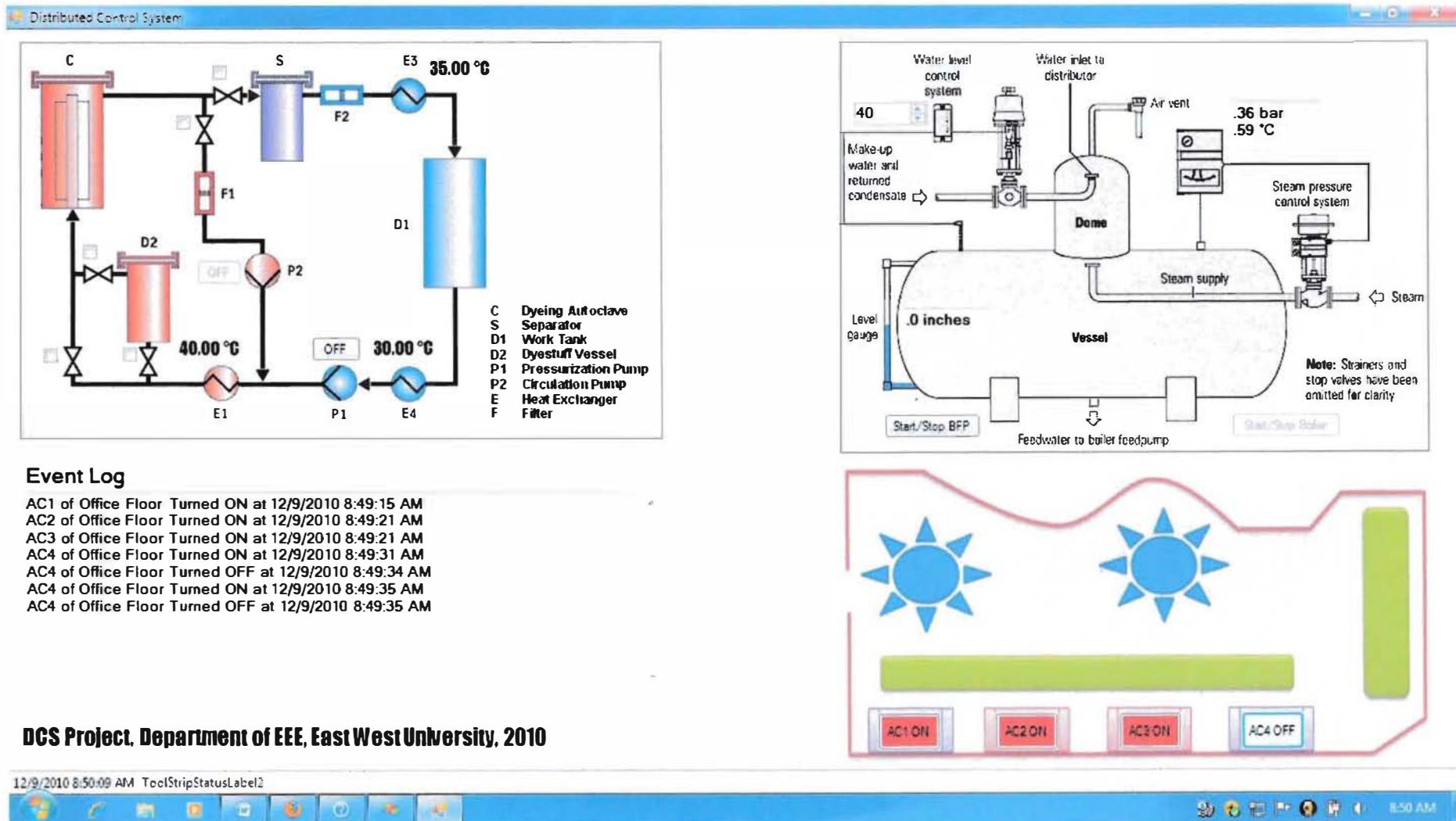


Figure : 3.1 GUI window of the DCS showing three processes, necessary GUI controls and Event Log.

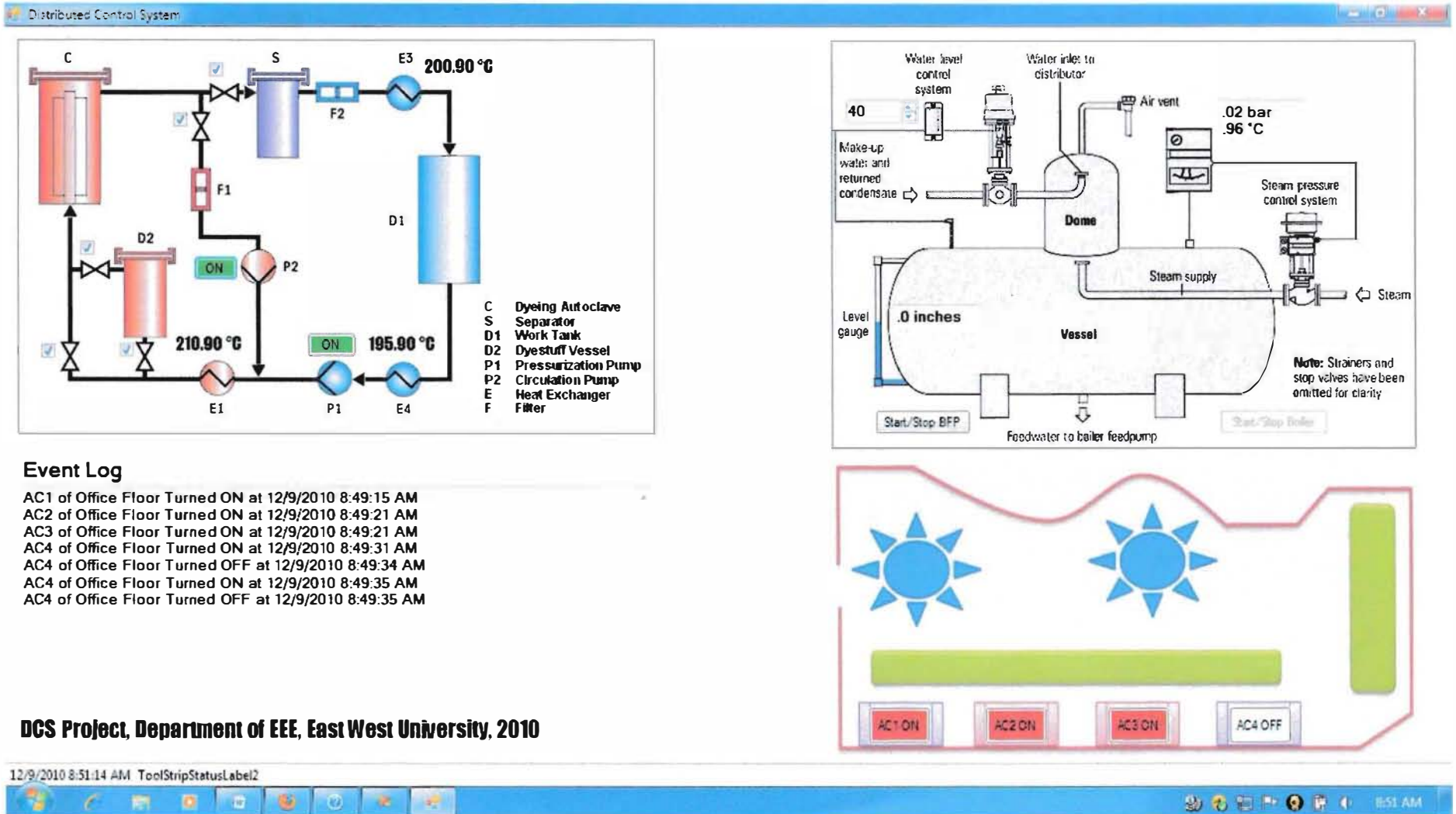
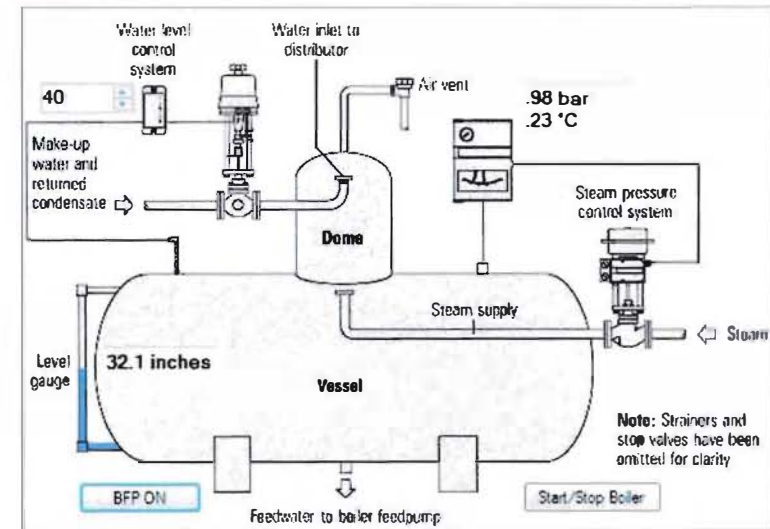
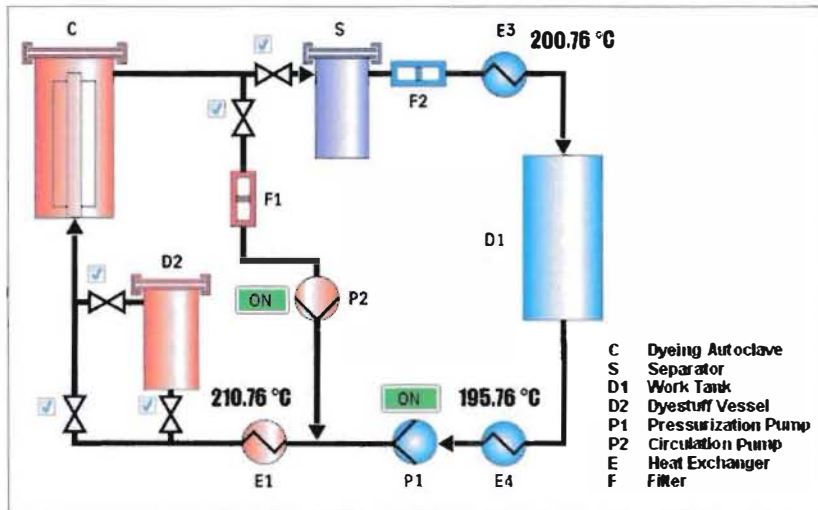
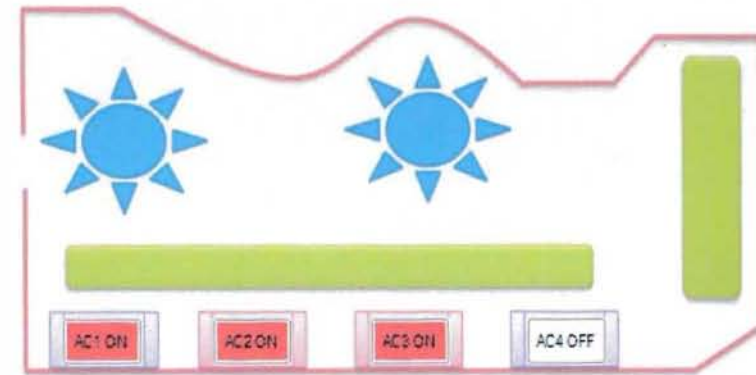


Figure: 3.2 DCS showing process 1 operation with all the valves closed and pumps ON.



Event Log

AC1 of Office Floor Turned ON at 12/9/2010 8:49:15 AM
 AC2 of Office Floor Turned ON at 12/9/2010 8:49:21 AM
 AC3 of Office Floor Turned ON at 12/9/2010 8:49:21 AM
 AC4 of Office Floor Turned ON at 12/9/2010 8:49:31 AM
 AC4 of Office Floor Turned OFF at 12/9/2010 8:49:34 AM
 AC4 of Office Floor Turned ON at 12/9/2010 8:49:35 AM
 AC4 of Office Floor Turned OFF at 12/9/2010 8:49:35 AM



DCS Project, Department of EEE, East West University, 2010

12/9/2010 8:51:49 AM ToolStripStatusLabel2



Figure : 3.3 DCS showing all three processes in operation.

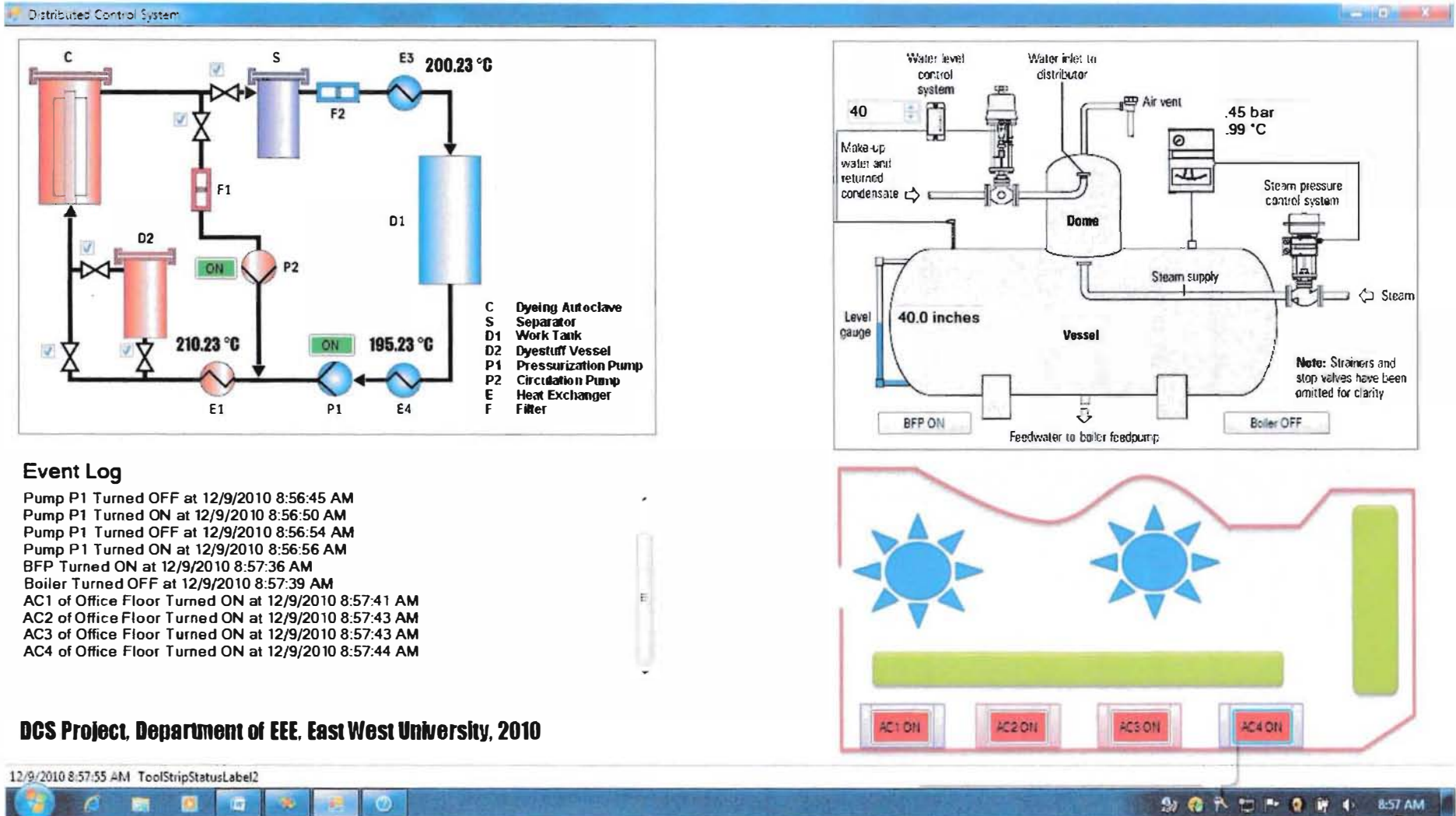


Figure: 3.4 DCS showing whole system operation with event logs for all processes.

3.3 Real Time Operation

All GUI controls in a windows form works asynchronously. The operating system assigns time for each GUI controls by its own. For real time operation of displaying system data (temperature, pressure etc.) continuously we need to ask the operating system for directing the processor to specific routines in regular intervals. We accomplished this real time works using a timer. On each tick, the timer requests the operating system to do its task by the processor. In our GUI form we used a timer "Timer_Display" to refresh all displays at an interval of 1 second. The Timer_Display_Tick subroutine do all the tasks of displaying the process data. The source code for the Timer_Display_Tick subroutine are shown below:

```
Private Sub Timer_Display_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer_Display.Tick
    If ButtonBFP.Text = "BFP ON" Then
        WaterLevel += 0.3
        If WaterLevel >= NumericUpDownWaterLevel.Value Then
            WaterLevel = NumericUpDownWaterLevel.Value
        End If
    End If
    If ButtonBFP.Text = "BFP OFF" Then
        WaterLevel -= 0.1
        If WaterLevel < 0 Then
            WaterLevel = 0
        End If
    End If
    TextBoxWaterLevel.Text = Format(WaterLevel, "##.0") & " inches"
    TextBoxSteam.Text = Format(Rnd(), "##.00") & " bar"
        & vbCrLf & Format(Rnd(), "##.00") & " °C"
    If CheckBoxValve2.CheckState = CheckState.Checked And
        CheckBoxValve4.CheckState = CheckState.Checked Then
        ButtonP2.Enabled = True
    Else
        ButtonP2.Enabled = False
        ButtonP2.BackColor = Me.BackColor
        ButtonP2.Text = "OFF"
        btnP2 = False
    End If
    ToolStripStatusLabel1.Text = Now
    If ButtonP1.Text = "ON" Then
        Celsius1 += 2
        If Celsius1 > 200 Then
            Celsius1 = 200 + Rnd()
        End If
    Else
        Celsius1 -= 1
        If Celsius1 <= 35 Then
            Celsius1 = 35
        End If
    End If
End Sub
```

```

End If
TextBoxE3.Text = Format(Celsius1, "##.00") & " °C"
TextBoxE4.Text = Format(Celsius1 - 5, "##.00") & " °C"
If ButtonP2.Text = "ON" Then
    Celsius2 = Celsius1 + 10
Else
    Celsius2 = Celsius1 + 5
End If
TextBoxE1.Text = Format(Celsius2, "##.00") & " °C"
End Sub

```

3.4 Direct Control of Process Equipments

All checkboxes and buttons are used to control process equipments. Due to constraint in the number of output lines in the USB-4711A module, we assigned output functions to only the buttons. Here is a list of the button assignment with the digital output lines.

Table 2 Process control assignment to digital output line of USB-4711A

Button Control	Digital Output Lines
Pump 1 of Process 1	DO0
Pump 2 of Process 1	DO1
Boiler Start/Stop of Process 2	DO2
Boiler Feed Pump (BFP) of Process 2	DO3
AC1 of Office Floor	DO4
AC2 of Office Floor	DO5
AC3 of Office Floor	DO6
AC4 of Office Floor	DO7

Once a process button is operated a text line appears in the status bar showing what data is communicating to the process equipment.

Chapter 4 Software Codes

The software developed in this work has some key functional parts:

- Subroutines for button clicked events
- Regular update of process information through timer tick events
- Data communication to the process equipment through USB 4711A digital I/O

4.1 Button Clicked Events

A user can click a button to operate the associate process equipment. The operation performed by the click event is visually available to the user by changing the button text and sometimes by changing button color. A typical subroutine that serves the ButtonP1.Click event is illustrated here.

```
Private Sub ButtonP1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonP1.Click
    bitP1 = Not bitP1
    If bitP1 = True Then
        ButtonP1.BackColor = Color.Green
        AxAdvDIO1.WriteDoChannel(1, 0) 'Pump 1, Process 1, D0
        '
        ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(0) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 1 sent to Pump P1 of
process 1"
        ButtonP1.Text = "ON"
        TextBox1.AppendText("Pump P1 Turned ON at " & Now & vbCrLf)
    Else
        ButtonP1.BackColor = Me.BackColor
        AxAdvDIO1.WriteDoChannel(0, 0) 'Pump 1, Process 1, D0
        '
        ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(0) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 0 sent to Pump P1 of
process 1"
        ButtonP1.Text = "OFF"
```

```

        TextBox1.AppendText("Pump P1 Turned OFF at " & Now &
vbCrLf)
    End If
End Sub

```

We defined bitP1 as a bit (Boolean variable) that stores the status of the pump 1 of process 1. Whenever, the user clicks the P1 button, the state of bitP1 changes. The pump is operated according to the state of bitP1. When bitP1 is assigned 1, the pump 1 is turned ON. When bitP1 is assigned to 0, pump 1 is turned OFF. For easy of visibility, the button background is also changed so that the user can confirm that his command is working. In a practical system, a feedback state (not incorporated in this design) from the process would be needed that would communicate to the GUI program through the USB4711A digital input lines.

4.2 Complete Program Codes

The whole program code written in VB.Net using Visual Studio 2008 is listed here to facilitate readers to understand and guide as how to write GUI program that would work interactively with process hardware.

4.2.1 DCS Main Form Codes

```

Imports System.Windows.Forms
Imports System.Drawing
Imports System.Drawing.Drawing2D

Public Class DCSForm
    Dim ONOFF_bit As Boolean = False
    Dim bitP1 As Boolean = False
    Dim bitP2 As Boolean = False
    Dim Celsius1 As Double = 0
    Dim Celsius2 As Double = 0
    Dim SteamTemp As Double = 0
    Dim SteamPressure As Double = 0
    Dim WaterLevel As Double = 0
    Dim btnP1 As Boolean = False
    Dim btnP2 As Boolean = False
    Dim BFPstate As Boolean = False

    Private Sub Timer_Display_Tick(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Timer_Display.Tick
        If ButtonBFP.Text = "BFP ON" Then
            WaterLevel += 0.3
            If WaterLevel >= NumericUpDownWaterLevel.Value Then

```

```

        WaterLevel = NumericUpDownWaterLevel.Value
    End If
End If
If ButtonBFP.Text = "BFP OFF" Then
    WaterLevel -= 0.1
    If WaterLevel < 0 Then
        WaterLevel = 0
    End If
End If
TextBoxWaterLevel.Text = Format(WaterLevel, "##.0") & " inches"
TextBoxSteam.Text = Format(10 + Rnd(), "##.00") & " bar"
                & vbCrLf & Format(200 + Rnd(), "##.00") & " °C"
If CheckBoxValve2.CheckState = CheckState.Checked And
CheckBoxValve4.CheckState = CheckState.Checked Then
    ButtonP2.Enabled = True
Else
    ButtonP2.Enabled = False
    ButtonP2.BackColor = Me.BackColor
    ButtonP2.Text = "OFF"
    btnP2 = False
End If
ToolStripStatusLabel1.Text = Now
If ButtonP1.Text = "ON" Then
    Celsius1 += 2
    If Celsius1 > 200 Then
        Celsius1 = 200 + Rnd()
    End If
Else
    Celsius1 -= 1
    If Celsius1 <= 35 Then
        Celsius1 = 35
    End If
End If
TextBoxE3.Text = Format(Celsius1, "##.00") & " °C"
TextBoxE4.Text = Format(Celsius1 - 5, "##.00") & " °C"
If ButtonP2.Text = "ON" Then
    Celsius2 = Celsius1 + 10
Else
    Celsius2 = Celsius1 + 5
End If
TextBoxE1.Text = Format(Celsius2, "##.00") & " °C"

```

```
End Sub
```

```

Private Sub DCSForm_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    AxAdvDIO1.ShowPropertyPages()
    Me.WindowState = FormWindowState.Maximized
End Sub

```

```

Private Sub PictureBox1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles PictureBox1.Click
    PictureBox1.BackColor = Color.DeepSkyBlue
End Sub

```

```

Private Sub ButtonP2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonP2.Click
    bitP2 = Not bitP2
    If bitP2 = True Then
        ButtonP2.BackColor = Color.Green
    End If
End Sub

```

```

        AxAdvDIO1.WriteDoChannel(1, 1) 'Pump 2, Process 1, D1
        ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(1) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 1 sent to Pump P2 of process
1"
        ButtonP2.Text = "ON"
        TextBox1.AppendText("Pump P2 Turned ON at " & Now & vbCrLf)
    Else
        ButtonP2.BackColor = Me.BackColor
        AxAdvDIO1.WriteDoChannel(0, 1) 'Pump 2, Process 1, D1
        ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(1) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 0 sent to Pump P2 of process
1"
        ButtonP2.Text = "OFF"
        TextBox1.AppendText("Pump P2 Turned OFF at " & Now & vbCrLf)
    End If
End Sub

Private Sub ButtonP1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonP1.Click
    bitP1 = Not bitP1
    If bitP1 = True Then
        ButtonP1.BackColor = Color.Green
        AxAdvDIO1.WriteDoChannel(1, 0) 'Pump 1, Process 1, D0
        ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(0) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 1 sent to Pump P1 of process
1"
        ButtonP1.Text = "ON"
        TextBox1.AppendText("Pump P1 Turned ON at " & Now & vbCrLf)
    Else
        ButtonP1.BackColor = Me.BackColor
        AxAdvDIO1.WriteDoChannel(0, 0) 'Pump 1, Process 1, D0
        ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(0) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 0 sent to Pump P1 of process
1"
        ButtonP1.Text = "OFF"
        TextBox1.AppendText("Pump P1 Turned OFF at " & Now & vbCrLf)
    End If
End Sub

Private Sub CheckBoxValve1_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CheckBoxValve1.CheckedChanged
    If CheckBoxValve1.Checked = True Then
        ButtonP1.Enabled = True
    Else
        ButtonP1.Enabled = False
        ButtonP1.BackColor = Me.BackColor
        ButtonP1.Text = "OFF"
    End If
End Sub

Private Sub ButtonBFP_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonBFP.Click
    If BFPstate = False Then
        BFPstate = True
        AxAdvDIO1.WriteDoChannel(1, 2) 'BFP, Process 2, D2
    End If
End Sub

```

```

    ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(2) & " sent to device"
    ToolStripStatusLabel2.Text = "Data 1 sent to BFP of process 2"
    ButtonBFP.Text = "BFP ON"
    TextBox1.AppendText("BFP Turned ON at " & Now & vbCrLf)
    ButtonGas.Enabled = True
Else
    BFPstate = False
    AxAdvDIO1.WriteDoChannel(0, 2) 'BFP, Process 2, D2
    ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(2) & " sent to device"
    ToolStripStatusLabel2.Text = "Data 0 sent to BFP of process 2"
    ButtonBFP.Text = "BFP OFF"
    TextBox1.AppendText("BFP Turned OFF at " & Now & vbCrLf)
    ButtonGas.Enabled = False
End If

End Sub

Private Sub ButtonGas_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonGas.Click
    If ButtonGas.Text = "Boiler OFF" Then
        AxAdvDIO1.WriteDoChannel(1, 3) 'Boiler Gas, Process 2, D3
        ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(3) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 1 sent to Boiler of process
2"

        ButtonGas.Text = "Boiler ON"
        TextBox1.AppendText("Boiler Turned ON at " & Now & vbCrLf)
    Else
        AxAdvDIO1.WriteDoChannel(0, 3) 'Boiler Gas, Process 2, D3
        ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(3) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 0 sent to Boiler of process
2"

        ButtonGas.Text = "Boiler OFF"
        TextBox1.AppendText("Boiler Turned OFF at " & Now & vbCrLf)
    End If

End Sub

Private Sub BtnAC1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnAC1.Click
    If BtnAC1.Text = "AC1 OFF" Then
        AxAdvDIO1.WriteDoChannel(1, 4) 'AC1, Office Floor, D4
        ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(4) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 1 sent to AC1 of Office
Floor"

        BtnAC1.Text = "AC1 ON"
        BtnAC1.BackColor = Color.Red
        TextBox1.AppendText("AC1 of Office Floor Turned ON at " & Now &
vbCrLf)
    Else
        BtnAC1.Text = "AC1 OFF"
        AxAdvDIO1.WriteDoChannel(0, 4) 'AC1, Office Floor, D4
        ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(4) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 0 sent to AC1 of Office
Floor"

        BtnAC1.BackColor = Me.BackColor

```

```

        TextBox1.AppendText("AC1 of Office Floor Turned OFF at " & Now
& vbCrLf)
        End If
    End Sub

    Private Sub BtnAC2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnAC2.Click
        If BtnAC2.Text = "AC2 OFF" Then
            AxAdvDIO1.WriteDoChannel(1, 5) 'AC2, Office Floor, D5
            ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(5) & " sent to device"
            ToolStripStatusLabel2.Text = "Data 1 sent to AC2 of Office
Floor"

            BtnAC2.Text = "AC2 ON"
            BtnAC2.BackColor = Color.Red
            TextBox1.AppendText("AC2 of Office Floor Turned ON at " & Now &
vbCrLf)
        Else
            AxAdvDIO1.WriteDoChannel(0, 5) 'AC2, Office Floor, D5
            ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(5) & " sent to device"
            ToolStripStatusLabel2.Text = "Data 0 sent to AC2 of Office
Floor"

            BtnAC2.Text = "AC2 OFF"
            BtnAC2.BackColor = Me.BackColor
            TextBox1.AppendText("AC2 of Office Floor Turned OFF at " & Now
& vbCrLf)
        End If
    End Sub

    Private Sub BtnAC3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnAC3.Click
        If BtnAC3.Text = "AC3 OFF" Then
            AxAdvDIO1.WriteDoChannel(1, 6) 'AC3, Office Floor, D6
            ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(6) & " sent to device"
            ToolStripStatusLabel2.Text = "Data 1 sent to AC3 of Office
Floor"

            BtnAC3.Text = "AC3 ON"
            BtnAC3.BackColor = Color.Red
            TextBox1.AppendText("AC3 of Office Floor Turned ON at " & Now &
vbCrLf)
        Else
            AxAdvDIO1.WriteDoChannel(0, 6) 'AC3, Office Floor, D6
            ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(6) & " sent to device"
            ToolStripStatusLabel2.Text = "Data 0 sent to AC3 of Office
Floor"

            BtnAC3.Text = "AC3 OFF"
            BtnAC3.BackColor = Me.BackColor
            TextBox1.AppendText("AC3 of Office Floor Turned OFF at " & Now
& vbCrLf)
        End If
    End Sub

    Private Sub BtnAC4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnAC4.Click
        If BtnAC4.Text = "AC4 OFF" Then
            AxAdvDIO1.WriteDoChannel(1, 7) 'AC4, Office Floor, D7
            ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(7) & " sent to device"

```

```

ToolStripStatusLabel2.Text = "Data 1 sent to AC4 of Office
Floor"
    BtnAC4.Text = "AC4 ON"
    BtnAC4.BackColor = Color.Red
    TextBox1.AppendText("AC4 of Office Floor Turned ON at " & Now &
vbCrLf)
    Else
        AxAdvDIO1.WriteDoChannel(0, 7) 'AC4, Office Floor, D7
        ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(7) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 0 sent to AC4 of Office
Floor"
        BtnAC4.Text = "AC4 OFF"
        BtnAC4.BackColor = Me.BackColor
        TextBox1.AppendText("AC4 of Office Floor Turned OFF at " & Now
& vbCrLf)
    End If
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    FormProcess1.Show()
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    FormProcess2.Show()
End Sub
End Class

```

4.2.2 Process 1 Form in a Separate Window

```

Public Class FormProcess1

    Private Sub Timer_Display_Tick(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Timer_Display.Tick

        If CheckBoxValve2.CheckState = CheckState.Checked And
CheckBoxValve4.CheckState = CheckState.Checked Then
            ButtonP2.Enabled = True
        Else
            ButtonP2.Enabled = False
            ButtonP2.BackColor = Me.BackColor
            ButtonP2.Text = "OFF"
            btnP2 = False
        End If
        ToolStripStatusLabel1.Text = Now
        If ButtonP1.Text = "ON" Then
            Celsius1 += 2
            If Celsius1 > 200 Then
                Celsius1 = 200 + Rnd()
            End If
        Else
            Celsius1 -= 1
            If Celsius1 <= 35 Then
                Celsius1 = 35
            End If
        End If
    End Sub
End Class

```

```

End If
TextBoxE3.Text = Format(Celsius1, "##.00") & " °C"
If ButtonP2.Text = "ON" Then
    Celsius2 = Celsius1 + 10
Else
    Celsius2 = Celsius1 + 5
End If
TextBoxE1.Text = Format(Celsius2, "##.00") & " °C"

DCSForm.TextBoxE1.Text = TextBoxE1.Text
DCSForm.TextBoxE3.Text = TextBoxE3.Text
DCSForm.CheckBoxValve1.Checked = CheckBoxValve1.CheckState
DCSForm.CheckBoxValve2.Checked = CheckBoxValve2.CheckState
DCSForm.CheckBoxValve3.Checked = CheckBoxValve3.CheckState
DCSForm.CheckBoxValve4.Checked = CheckBoxValve4.CheckState
DCSForm.CheckBoxValve5.Checked = CheckBoxValve5.CheckState
End Sub
Private Sub ButtonP2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonP2.Click
    bitP2 = Not bitP2
    If bitP2 = True Then
        ButtonP2.BackColor = Color.Green
        DCSForm.AxAdvDIO1.WriteDoChannel(1, 1) 'Pump 2, Process 1, D1
        '    ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(1) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 1 sent to Pump P2 of process
1"

        ButtonP2.Text = "ON"
        DCSForm.TextBox1.AppendText("Pump P2 Turned ON at " & Now &
vbCrLf)
    Else
        ButtonP2.BackColor = Me.BackColor
        DCSForm.AxAdvDIO1.WriteDoChannel(0, 1) 'Pump 2, Process 1, D1
        '    ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(1) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 0 sent to Pump P2 of process
1"

        ButtonP2.Text = "OFF"
        DCSForm.TextBox1.AppendText("Pump P2 Turned OFF at " & Now &
vbCrLf)
    End If
    DCSForm.ButtonP2.Text = ButtonP2.Text
    DCSForm.ButtonP2.BackColor = ButtonP2.BackColor

End Sub

Private Sub ButtonP1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonP1.Click
    bitP1 = Not bitP1
    If bitP1 = True Then
        ButtonP1.BackColor = Color.Green
        DCSForm.AxAdvDIO1.WriteDoChannel(1, 0) 'Pump 1, Process 1, D0
        '    ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(0) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 1 sent to Pump P1 of process
1"

        ButtonP1.Text = "ON"
        DCSForm.TextBox1.AppendText("Pump P1 Turned ON at " & Now &
vbCrLf)
    Else

```



```

        ButtonP1.BackColor = Me.BackColor
        DCSForm.AxAdvDIO1.WriteDoChannel(0, 0) 'Pump 1, Process 1, DO
        '   ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(0) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 0 sent to P mp P1 of process
1"
        ButtonP1.Text = "OFF"
        DCSForm.TextBox1.AppendText("Pump P1 Turned OFF at " & Now &
vbCrLf)
    End If
    DCSForm.ButtonP1.Text = ButtonP1.Text
    DCSForm.ButtonP1.BackColor = ButtonP1.BackColor
End Sub

Private Sub CheckBoxValve1_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CheckBoxValve1.CheckedChanged
    If CheckBoxValve1.Checked = True Then
        ButtonP1.Enabled = True
    Else
        ButtonP1.Enabled = False
        ButtonP1.BackColor = Me.BackColor
        ButtonP1.Text = "OFF"
    End If
End Sub
End Class

```



4.2.3 Process 2 Program in a Separate Window

```

Public Class FormProcess2

    Private Sub Timer_Display_Tick(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Timer_Display.Tick
        If ButtonBFP.Text = "BFP ON" Then
            WaterLevel += 0.3
            If WaterLevel >= NumericUpDownWaterLevel.Value Then
                WaterLevel = NumericUpDownWaterLevel.Value
            End If
        End If
        If ButtonBFP.Text = "BFP OFF" Then
            WaterLevel -= 0.1
            If WaterLevel < 0 Then
                WaterLevel = 0
            End If
        End If
        TextBoxWaterLevel.Text = Format(WaterLevel, "##.0") & " inches"
        TextBoxSteam.Text = Format(10 + Rnd(), "##.00") & " bar"
            & vbCrLf & Format(200 + Rnd(), "##.00") & " °C"
        DCSForm.TextBoxWaterLevel.Text = TextBoxWaterLevel.Text
        DCSForm.TextBoxSteam.Text = TextBoxSteam.Text
        DCSForm.NumericUpDownWaterLevel.Value =
NumericUpDownWaterLevel.Value

        ToolStripStatusLabel1.Text = Now
    End Sub

```

```

Private Sub ButtonBFP_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonBFP.Click
    If BFPstate = False Then
        BFPstate = True
        DCSForm.AxAdvDIO1.WriteDoChannel(1, 2) 'BFP, Process 2, D2
        ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(2) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 1 sent to BFP of process 2"
        ButtonBFP.Text = "BFP ON"
        DCSForm.TextBox1.AppendText("BFP Turned ON at " & Now & vbCrLf)
        ButtonGas.Enabled = True
    Else
        BFPstate = False
        DCSForm.AxAdvDIO1.WriteDoChannel(0, 2) 'BFP, Process 2, D2
        ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(2) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 0 sent to BFP of process 2"
        ButtonBFP.Text = "BFP OFF"
        DCSForm.TextBox1.AppendText("BFP Turned OFF at " & Now &
vbCrLf)
        ButtonGas.Enabled = False
    End If
    DCSForm.ButtonBFP.Text = ButtonBFP.Text
    DCSForm.ButtonBFP.BackColor = ButtonBFP.BackColor
End Sub

Private Sub ButtonGas_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButtonGas.Click
    If ButtonGas.Text = "Boiler OFF" Then
        DCSForm.AxAdvDIO1.WriteDoChannel(1, 3) 'Boiler Gas, Process 2,
D3
        ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(3) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 1 sent to Boiler of process
2"
        ButtonGas.Text = "Boiler ON"
        DCSForm.TextBox1.AppendText("Boiler Turned ON at " & Now &
vbCrLf)
    Else
        DCSForm.AxAdvDIO1.WriteDoChannel(0, 3) 'Boiler Gas, Process 2,
D3
        ' ToolStripStatusLabel2.Text = "Data " &
AxAdvDIO1.ReadDoChannel(3) & " sent to device"
        ToolStripStatusLabel2.Text = "Data 0 sent to Boiler of process
2"
        ButtonGas.Text = "Boiler OFF"
        DCSForm.TextBox1.AppendText("Boiler Turned OFF at " & Now &
vbCrLf)
    End If
    DCSForm.ButtonGas.Text = ButtonGas.Text
    DCSForm.ButtonGas.BackColor = ButtonGas.BackColor
End Sub

End Class

```

4.2.4 Global Variable Declarations

```
Module Module1
    Public bitP1 As Boolean = False
    Public bitP2 As Boolean = False
    Public Celsius1 As Double = 0
    Public Celsius2 As Double = 0
    Public SteamTemp As Double = 0
    Public SteamPressure As Double = 0
    Public WaterLevel As Double = 0
    Public btnP1 As Boolean = False
    Public btnP2 As Boolean = False
    Public BFPstate As Boolean = False
End Module
```

Chapter 5 Results



5.1 DCS Software

Few snapshots of the DCS are presented in this chapter that contains all the functionalities. In a real DCS, all the processes are not displayed in a single window. Rather, the processes are linked to controls in the main window, where each processes are opened in separate windows.

The DCS operation in integrated form and processes operation from separate windows are shown in Figs. 5.1-5.4.

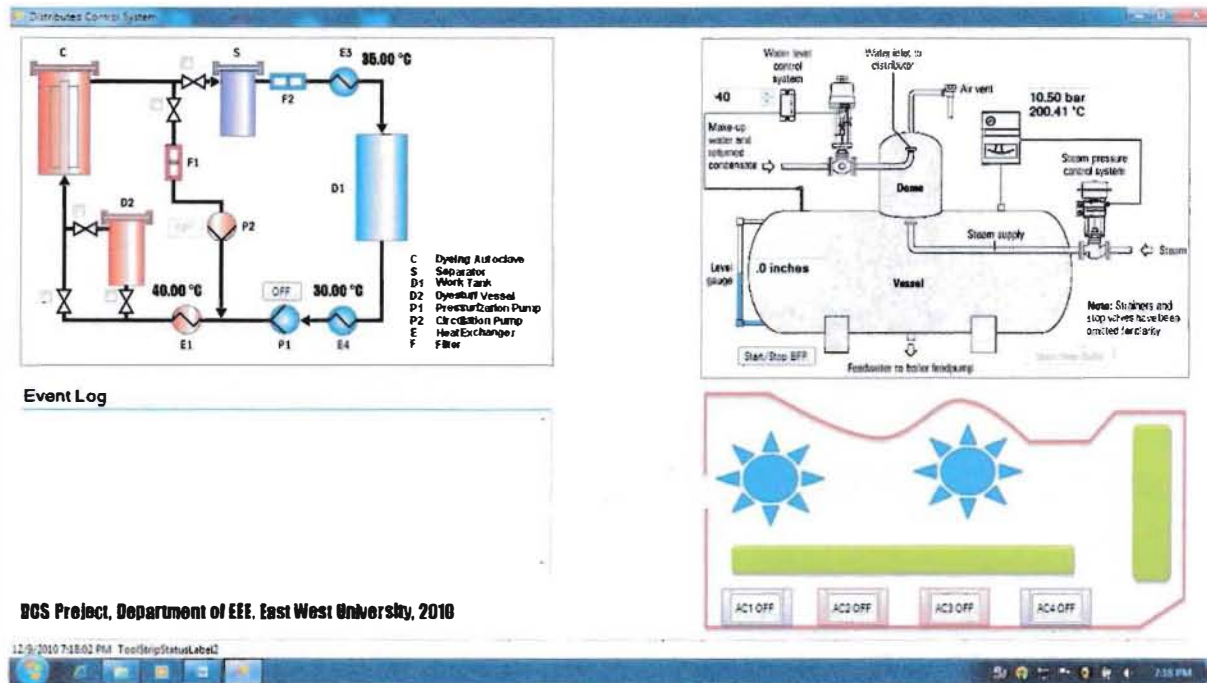


Figure 5.1 All processes in the DCS shown in same window.

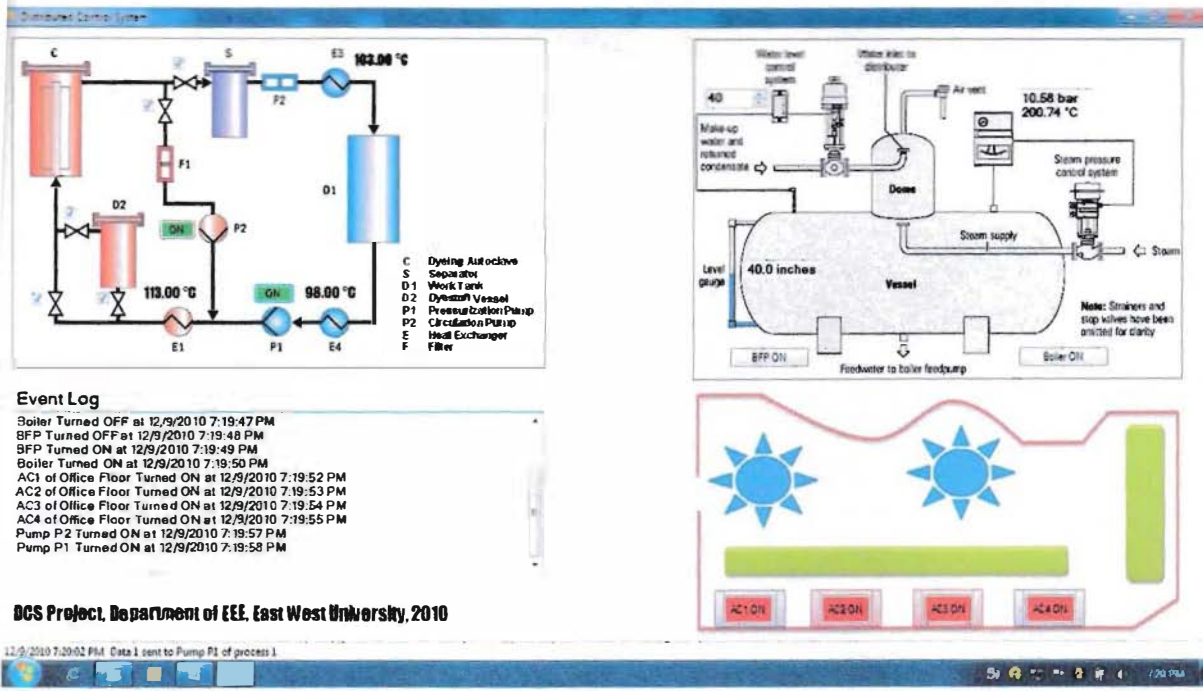


Figure 5.2 DCS operation showing event logs.

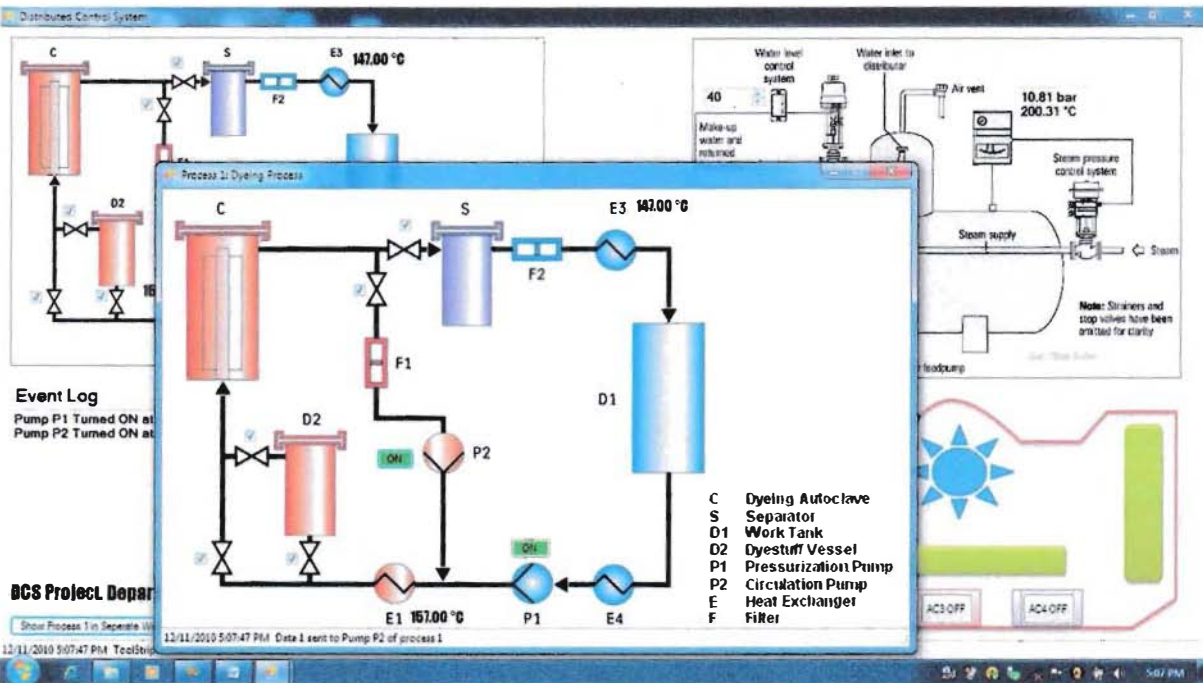
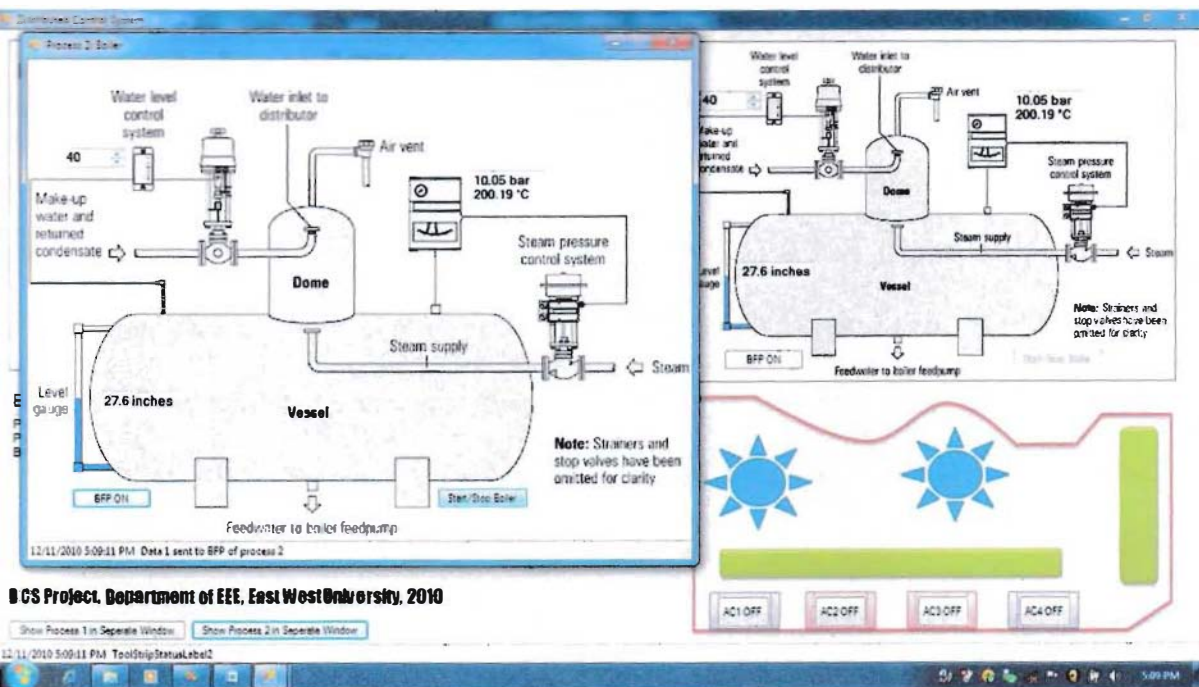


Figure 5.3 DCS showing process 1 in a separate window linked to the main GUI.



DCS Project, Department of EEE, East West University, 2010

Figure 5.4 DCS showing process 2 opened in a separate window.

5.2 DCS Interface Hardware

The DCS interface hardware is shown in Fig. 5.5 and 5.6 respectively. The 12V DC power supply is given from a laboratory dc power supply. The hardware boards contain relay interface as well as other electronic devices.

5.3 System Operation

The system operation of the DCS is tested in the lab. The DCS GUI is run in a laptop while the Advantech USB4711A is interfaced through the USB port. All combinations of control operations are tested and the DCS is found to operate the relays successfully without any problem. The digital output from the USB4711A cannot be conductively connected to the relay circuit as high voltage power backflow may damage the DAQ module. The digital output signal is isolated using 4N35 optocoupler and is amplified using BD135 transistor. The relay operates from a 12V dc supply while the DAQ module is operated from USB power that is 5V only. The DLL file AxAdvDIO1.WriteDoChannel(1,0) writes 1 to DO0

channel and AxAdvDIO1. WriteDoChannel(0,1) writes 0 to DO0 channel. Relay controls are also incorporated in the program where the user can individually select a relay and turn it ON or OFF using mouse clicks on the button. When button ON then Digital output from DO0 is 1 and when button OFF then Digital output from DO0 is 0. This Digital output goes to the Optocoupler 4N35 in relay control circuit. Optocoupler 4N35 gets 1 then relay ON and starts the connecting system operation, Optocoupler 4N35 gets 0 then relay OFF and stops the connecting system operation. Relay ON/OFF system is the same for DO0 to DO7 Channel. If boiler, Dyeing process and air conditioners connect with relay output then it is possible to stop or start all three systems and monitor temperature of Dyeing machine, control water level of boiler and also monitor boiler temperature and pressure by operating relay control circuit in Direct Control of Process Equipment in a DCS.

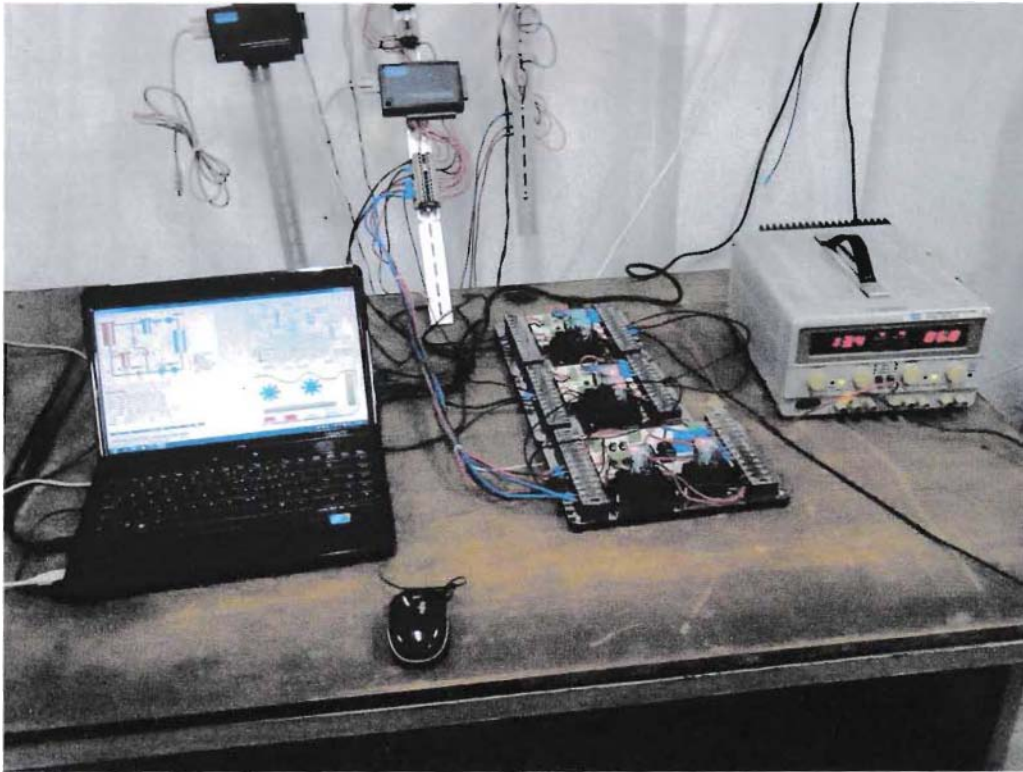


Figure 5.5 Experimental setup of the DCS hardware for direct equipment control.

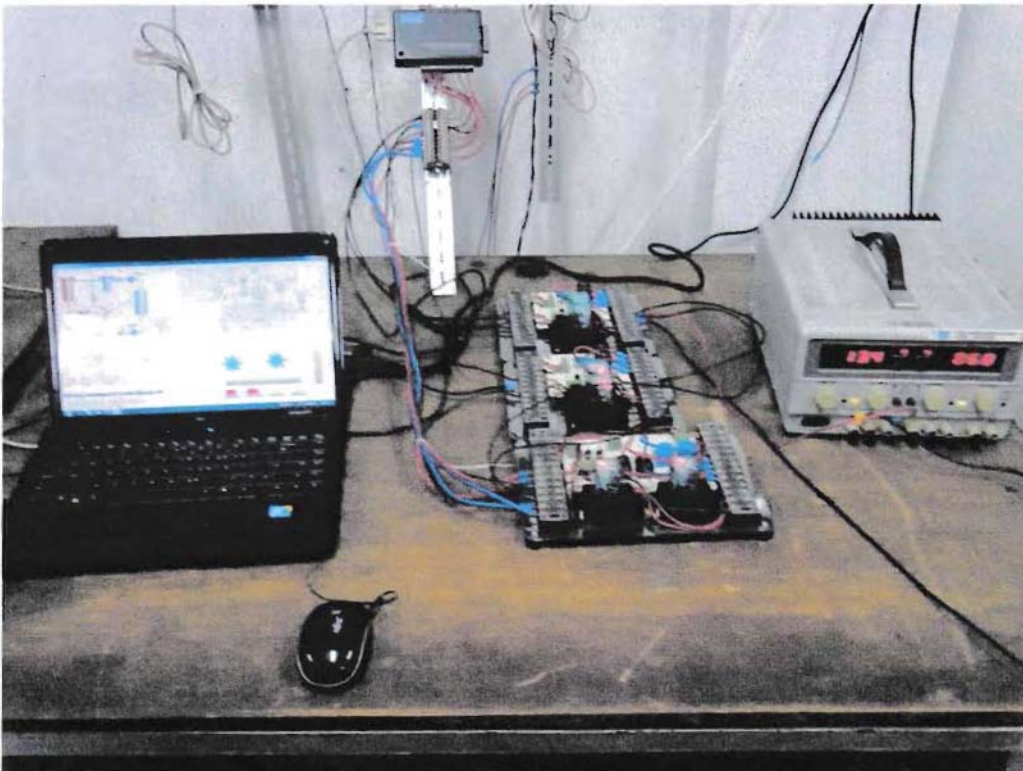


Figure 5.6 DCS running from a laptop and interfaced to relay board.

Chapter 6 Conclusions

6.1 Conclusions

In this project we developed a GUI interface for direct equipment control from a DCS. Detail procedures of the work are described in this report that would guide future works based on this project. The developed program can be tested in simulated mode by choosing the Advantech Demo I/O when the program starts. A user needs to install the Advantech DAQPro software and include the “Advantech Demo Board” by running the “Advantech Device Manager”. The developed program works in real mode with DAQ-USB-4711A board as well. A user has to choose the board from the list once the program starts.

6.2 Future Works

We tested our program and it is found to work satisfactorily, however, we did not implement it in a real environment where the relay output is set to control pumps, motors, AC etc. Future work may be focused on implementing this strategy in a real environment. Further, development of a fully fledged mini DCS with networking, process controller and direct controls may be undertaken as a major University Project.



Bibliography

- 1] Stout, T. M. and Williams, T. J. "Pioneering Work in the Field of Computer Process Control." *IEEE Annals of the History of Computing* 17 (1).1995
- 2] Integrated Production Control System CENTUM VP System Overview (Vnet/IP Edition), Yokogawa Electric Corporation. 2010
- 3] INFI 90, <http://www.abb.com/controlsystems>
- 4] DCI-4000, <http://www.abb.com/product/us>
- 5] Foundation Fieldbus, <http://www.fieldbus.org>
- 6] ABB System 800xA, <http://www.abb.com/product/us>
- 7] Emerson Process Management, <http://easydeltav.com>
- 8] Siemens, <http://www.pcs7.com>
- 9] Simatic PCS 7, <http://www.pcs.khe.siemens.com>
- 10] Yamatake's azbil, <http://www.azbil.com>