รายการอ้างอิง

1. IEEE recommended practice for master/remote supervisory control and data acquisition (SCADA) communications ฉบับภาษาไทย. แปลและเรียบเรียงโดย สมาคมสถาบันวิศวกรไฟฟ้าและอิเล็กส์ทรอนิกส์แห่งประเทศไทย (IEEE Thailand Section) , กรุงเทพฯ : การไฟ ฟ้าส่วนภูมิภาค, 2539.
2. IEEE Standard Definition , Specification and Analysis of Systems Used For Supervisory Control , Data Acquisition and Automatic Control (IEEE Std. 37.1 - 1994).
3. Chen Qizhi., Qian Qinquan..The research of UNIX platform for SCADA. Power Engineering Society Winter Meeting, 2000. IEEE pp. 2041-2045 vol.3, 23-27 Jan. 2000.
4. Thomas Hergenhanhahn. Visual is a set of programs to control, operate and monitor industrial machinery from a computer screen,on a local computer or over intra-/internet [Online]. Available from: http://sourceforge.net [2002, June 8].
5. Unel G., Ambrosini G., Conka T., Crone G., Fernanedes A., Francis D., Joos M., Lehmann G., Lopez J., Mailov A., Mapelli L., Mornacchi G., Niculescu M., Petersen J., Tremblet L., Veneziano S., Wildish T., Yasu Y.. Using Linux PCs in DAQ applications , Real Time Conference, 1999. Santa Fe 1999. 11th IEEE NPSS, pp. 73 –77, 14-18 June 1999.
6. B.Qiu and H.B.Gooi , "Web-Based SCADA Display Systems (WSDS) for Access via Internet", IEEE Transactions on Power Systems , vol. 15.,pp. 681-686, May 2000.
7. Stuart A. Boyer. SCADA Supervisory Control and Data Acquisition. North Carolina : Iliad Engineering, 2536.
8. Micheal R. Sweet. Serial Programming Guide for POSIX Operating. GNU Free Documentation License. 1999.
9. Serial Programming Example. [Online] Available from: <URL:http://www.hi-ho.ne.jp/a-enomoto/kylixmemo/serial/hmcomm-0.1.tgz> [2003, September 1].

10. ระบบ SCADA โครงการประตูระบายน้ำอุทกวิภาชประสิทธิ , โครงการส่งน้ำและบำรุงรักษาปากพนัง สำนักชลประทานที่ 15 กรมชลประธาร [Online]. Available from: http://203.150.73.21/rid11/ppn/me/me32.htm [2002, July 5].

11. ระบบป้องกันน้ำท่วมกรุงเทพมหานคร (BMA Flood Control Center system),สำนักงานระบายน้ำ กรุงเทพมหานคร[Online]. Available from: http://tiwrm.hpcc.nectec.or.th/SURVEY/BMA/fcc.html [2002, July 5].

12. Data Logging Software Package for AQSTAION, DARWIN for Windows 95/98/NT. Bulletin 04D05C01 – 00E., YOKOKAWA ELECTRIC CORPORATION.

13. Charles Calvert, Marjorie Calvert, John Kaster, Bob Swart.Borland Kylix Delveloper's Guide.สำนักพิมพ์ SAMS, 2545.

14. Alessandro Rubini & Jonathan Corbet .LINUX DEVICE DRIVERS. Cambridge : SPD O'REILLY, 2543.

15. ภัทระ เกียรติเสวี. สร้างอินเทอร์เนตเซิร์ฟเวอร์ด้วย LINUX. พิมพ์ครั้งที่ 2. กรุงเทพมหานคร : สำนักพิมพ์ซีเอด, 2545.

16. ก่อกิจ วีระอาชากุล. ติดตั้งและปรับแต่งเซิร์ฟเวอร์ LINUX . พิมพ์ครั้งที่ 2. กรุงเทพมหานคร : สำนักพิมพ์ Professional Series, 2543

17. ประพนธ์ อัศวภาณุวัฒน์. Delphi Episode II เทคนิคและการพัฒนาโปรแกรมด้วยเดลไฟ. กรุงเทพมหานคร : สำนักพิมพ์ซีเอด, 2543.

18. AEG SHNIDER AUTOMATION. Modicon Modbus Network Planning and Installation Guide. Version 3.0, 1996, 890 USE 10000. U.S.A..

19. Kylix 3 Enterprise Trial and Companion Tools For Kylix. [Online]. Available from: http://www.borland.com, http://sourceforge.net, http://www.iocomp.com. [2002, September 5].

20. MySQL Database Server. [Online] Available from: http://www.mysql.com [2002, October 18].

21. Modbus Protocol. [Online] Available from: http://www.modicon.com/techpubs/intr7.html [2003, January 15].

22. AI210 Protocol User Manual. [Online]. Available from: http://www.wisco.co.th. [2002, September 5].

ภาคผนวก

ภาคผนวก ก
รายละเอียดซอฟต์แวร์การสื่อสารพอร์ตอนุกรมและการอ่านข้อมูลจากสถานีปลายทางระยะโดยการโพลลิ่ง (Polling) ของระบบสกาดา

1. ตัวอย่างการสร้างอุปกรณ์ทางซอฟต์แวร์สำหรับการสื่อสารทางพอร์ตอนุกรมของระบบปฏิบัติการลีนุกซ์

```
unit GComm;

interface

uses Classes, SysUtils, Libc;

type
    TGRxThread =class;
    TGComm =class;
    TGRxEvent =procedure(Sender: TGComm; nchars: Integer)of object;
    TGComm =class(TComponent)
private
    fd: Integer;
    oldtio: termios;
    have_oldtio: boolean;
    FGRxThread: TGRxThread;
    FOnReceive: TGRxEvent;
    FDevice: string;

   procedure DoRxEvent;
   procedure SetActive(const Value: boolean);
   function GetActive: boolean;
public
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;

   procedure Open;
   procedure Close;

   function Read(var buff; size: Integer): Integer;
   procedure ReadStr(var buff: string);
   function Write(const buff; size: Integer): Integer;
   procedure WriteStr(const buff: string);

property Active: boolean read GetActive write SetActive;

published
   property Device: string read FDevice write FDevice;
   property OnReceive: TGRxEvent read FOnReceive write FOnReceive;
end;

   TGRxThread =class(TThread)
   private
   FOwner: TGComm;

protected
procedure Execute; override;
```

```
procedure InvokeRxEvent;

public
  constructor Create(AOwner: TGComm);
end;

procedure Register;

implementation

uses KernelIoctl;

const
       devfile = '/dev/ttyS0';

{ TGComm }

 constructor TGComm.Create(AOwner: TComponent);
    begin
      inherited;

      fd :=-1; //not opened
      have_oldtio := false;
      FDevice := devfile;
    end;

 destructor TGComm.Destroy;
    begin
      if fd >=0 then
         begin
            Close;
         end;
      inherited;
    end;

 procedure TGComm.Open;
 var  rc: Integer;
      newtio: termios;
 begin
     if fd >=0 then
     begin
        //already opened...
        Close;
     end;

     fd := Libc.open(PChar(FDevice), O_RDWR or O_NOCTTY);
     if fd < 0 then
       begin
         perror('open: '); //TODO: throw 'cannot open device' exception
       end
       else  begin
         rc := tcgetattr(fd, oldtio);
         if rc =0 then
             begin
```

```
                have_oldtio := true;
                newtio := oldtio;

            //always set
             newtio.c_cflag := newtio.c_cflag or CLOCAL or CREAD;
            //19200bps, 8N1
             newtio.c_cflag := newtio.c_cflag and not PARENB;
             newtio.c_cflag := newtio.c_cflag and not CSTOPB;
             newtio.c_cflag := newtio.c_cflag and not CSIZE;
             newtio.c_cflag := newtio.c_cflag or CS8;
            //TODO: implement baud rate setting
             cfsetispeed(newtio, B19200);
             cfsetospeed(newtio, B19200);
            //TODO: implement other port setting
            //disable RTSCTS
            newtio.c_cflag := newtio.c_cflag and not CRTSCTS;
            //raw input
             newtio.c_lflag := newtio.c_lflag and not (ICANON or ECHO or
             ECHOE or ISIG);
             newtio.c_iflag := newtio.c_iflag and not ICRNL;
            //disable parity check
             newtio.c_iflag := newtio.c_iflag and not (INPCK or ISTRIP);
            //enable software flow control
             newtio.c_iflag := newtio.c_iflag or IXON or IXOFF or IXANY;
            //raw output
             newtio.c_oflag := newtio.c_oflag and not OPOST;
            //disable character timer
             newtio.c_cc[VTIME] := #0;
             newtio.c_cc[VMIN] := #0;
            //set attributes
             tcflush(fd, TCIFLUSH);
             tcsetattr(fd, TCSANOW, newtio);
            //start Rx thread
             FGRxThread := TGRxThread.Create(Self);
         end
      else begin
             have_oldtio := false;
           end;
      end;
end;

procedure TGComm.Close;
begin
      if fd >= 0 then
      begin
         if have_oldtio then
         begin
```

```
            tcsetattr(fd, TCSANOW, oldtio);
         end;
      if assigned(FGRxThread)then
         begin
          FGRxThread.Terminate;
          FGRxThread := nil;
         end;
        __close(fd);
        fd := -1;
        have_oldtio := false;
       end;
end;

procedure TGComm.DoRxEvent;
var
      nchars: Integer;
      rs, rc: Integer;
begin
      if assigned(FOnReceive)then
      begin
         rc := ioctl(fd, FIONREAD, @nchars);
         if rc <> 0 then chars := 0;
         FOnReceive(Self, nchars);
      end;
end;

function TGComm.Read(var buff; size: Integer): Integer;
begin
     Result := __read(fd, buff, size);
end;

procedure TGComm.ReadStr(var buff: string);
type
      CharArray = array[0..MAXINT div 2]of Char;
      PCharArray = ^CharArray;
var
      blen: Integer;
      rc: Integer;
     pstr: PCharArray;
begin
      buff := '';
      rc := ioctl(fd, FIONREAD, @blen);
      if rc =0 then
      begin
         SetLength(buff, blen);
         pstr := @buff[1];
         blen := __read(fd, pstr^, blen);
         SetLength(buff, blen);
      end;
end;
```

```
function TGComm.Write(const buff; size: Integer): Integer;
begin
      // TODO: check send buffer size
      Result := __write(fd, buff, size);
end;

procedure TGComm.WriteStr(const buff: string);
type
  CharArray = array[0..MAXINT div 2] of Char;
  PCharArray = ^CharArray;
var
  pstr: PCharArray;
begin
  pstr := @buff[1];
  __write(fd, pstr^, Length(buff));
end;

procedure TGComm.SetActive(const Value: boolean);
begin
  if (fd < 0) and Value then Open
  else if (fd >= 0) and not Value then Close;
end;

function TGComm.GetActive: boolean;
begin
  if fd < 0 then Result := false
  else Result := true;
end;

{ TGRxThread }
constructor TGRxThread.Create(AOwner: TGComm);
    begin
        inherited Create(True);
        // dprintf(stderr, 'Starting GRxThread...'#$D#$A);
        FOwner := AOwner;
        FreeOnTerminate := true;
        Resume;
    end;

procedure TGRxThread.Execute;
var maxfd: Integer;
    readfds: TFdSet;
    timeout: TTimeVal;
    fd: Integer;
    rc: Integer;
begin
    fd := FOwner.fd;
    maxfd := fd + 1;
    // dprintf(stderr, 'Starting Execute...'#$D#$A);
```

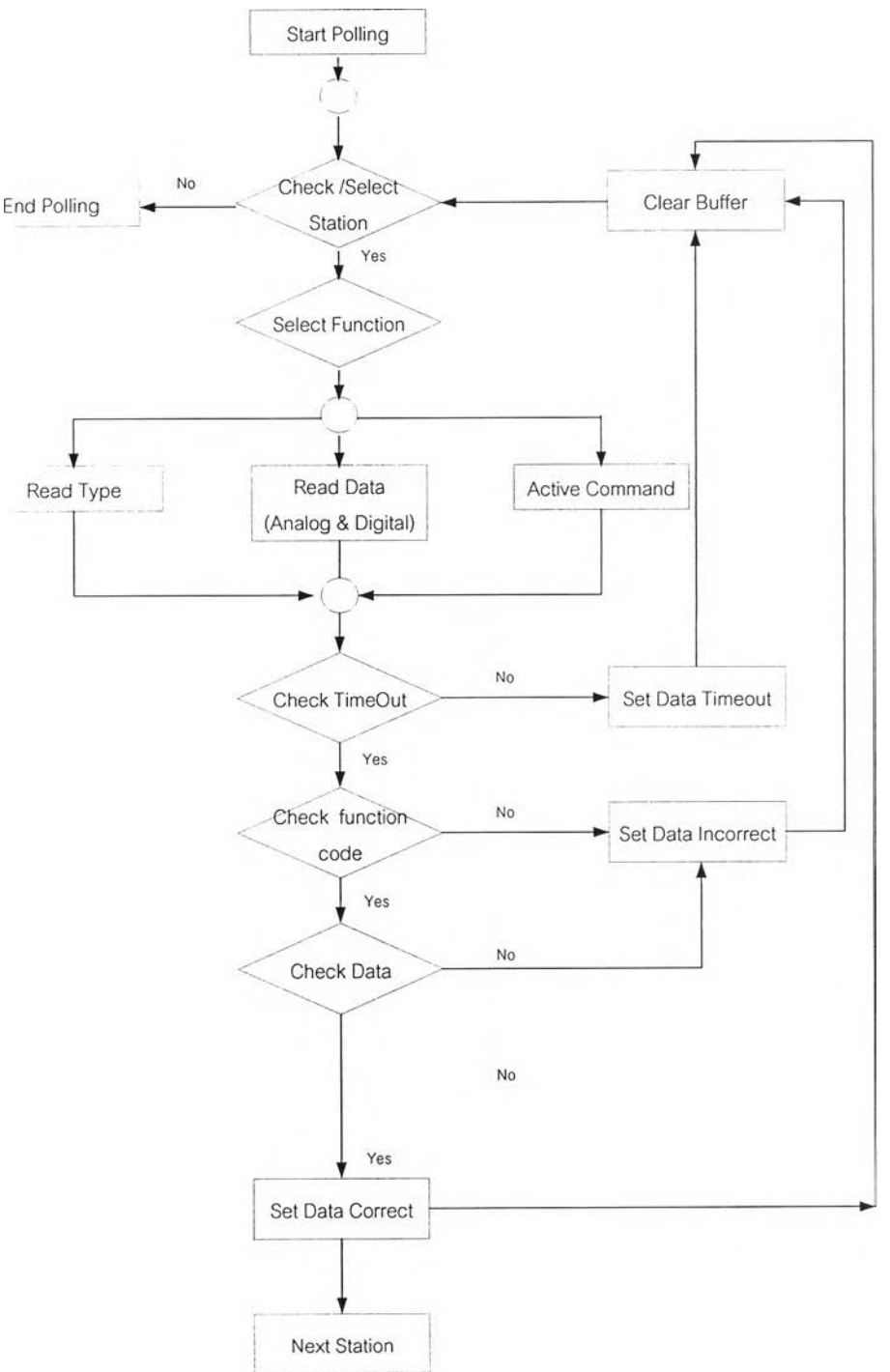
```
    while not Terminated do
    begin
       FD_SET(fd, readfds);
       timeout.tv_sec := 1;
       timeout.tv_usec := 0;
       //block thread...
       rc := select(maxfd, @readfds, nil, nil, @timeout);
       if Terminated then
       Exit;
       // dprintf(stderr, '  in select(2) loop'#$D#$A);
      if FD_ISSET(fd, readfds) then
       begin
       Synchronize(InvokeRxEvent);
       end;
  end;
end;

procedure TGRxThread.InvokeRxEvent;
begin
      FOwner.DoRxEvent;
end;

procedure Register;
begin
     RegisterComponents('Samples', [TGComm]);
end;

end.
```

2. ตัวอย่างซอฟต์แวร์สำหรับการการอ่านข้อมูลจากสถานีปลายทางระยะไกลโดยการโพลลิ่ง (Polling) ของระบบสกาดาโดยโพรโตคอลมอส์ดบัสแอสกี (MODBUS ASCII)



รูปที่ 1 แสดงโครงสร้างการโพลลิ่งข้อมูลโดยใช้โพรโตคอลมอส์ดบัสแอสกี (MODBUS ASCII)

```
unit ModBusPolling;

interface
uses
  SysUtils, Types, Classes, Variants, QGraphics, QControls, QForms,
QDialogs,
  QTypes, AxMisc, AxPort,Station,IdGlobal,math;

{**********เพื่อให้ง่ายในตัวอย่างนี้ใช้ Component การสื่อสารแบบอนุกรม (Serial) ของ
  TurboPower Async Professional CLX 1.01 Download ฟรีที่ Sourceforge.net
 **********}

type
    TPolling =class (TComponent)
    private
      FBuf: String;
      FFinishFlag: Boolean;
      FActiveCommand2:Boolean;
      procedure ApxComPort1TriggerAvail(CP: TObject; Count: Word);
      function CheckTimeOut(TimeOut: Cardinal):Boolean;
      procedure DataIn();
      procedure StationTimeOut(Sta: TStation);
      procedure SetAnalogToStation(Station: TStation;Data: String);
      procedure SetDigitalToStation(Station: TStation;Data: String);
      procedure SetTypeAnalog(Station: TStation; Data: String);
      procedure DataIncorrect(Sta: TStation);
    public
      ApxComPort1: TApxComPort;
      TimeOut: Cardinal;

      constructor Create(AOwner: TComponent);override;
      destructor Destroy;override;

      procedure Poll(var Station: Array of TStation);
      Procedure ScanStation(var Station: array of TStation);
      function
Ative_Command2(Number_station:Integer;Ch:String;Act:string):Boolean;
      procedure ReadAnalogType(var Station: array of TStation);
    end;
implementation
{ TPolling }

constructor TPolling.Create(AOwner: TComponent);
begin
  inherited;
  ApxComPort1:=TApxComPort.Create(self); //Create Asyn Pro Component
  ApxComPort1.OnTriggerAvail:=ApxComPort1TriggerAvail; //Set event
  ApxComPort1.ComNumber:=1;  //Set Default Port Com1
  Timeout:=1000;  //Set Default Time Out
end;
```

```
destructor TPolling.Destroy;
begin
  ApxComPort1.Open:=False; //Close Comport
  ApxComPort1.Destroy; //Destroy Ayn Pro Component
  inherited;
end;

procedure TPolling.ApxComPort1TriggerAvail(CP: TObject; Count: Word);
var //PpdComPort's OnTriggerAvail Function
   i: Integer;
   c: Char;
begin
   for i:=1 to count do //Read data to buffer
   begin
      c:=ApxComPort1.GetChar;
      FBuf:=FBuf+c;
      if c=#13 then DataIn;  //Finish Data Stream
   end;
end;

function TPolling.CheckTimeOut(TimeOut: Cardinal): Boolean;
var
   StartTime: Cardinal;
   StopTime: Cardinal;
begin
     StartTime:= GetTickCount;
     repeat
           Application.ProcessMessages;
           StopTime:=GetTickCount;
     until ((FFinishFlag)or(StopTime-StartTime>TimeOut));
//Exit when Timeout or have data
     Result:=not FFinishFlag;
end;

procedure TPolling.DataIn;
begin
   FFinishFlag:=true; //Clear flag when data complete
end;

procedure TPolling.Poll(var Station: array of TStation);
var
   i: Integer;
   CheckSum,CS1,CS2:byte;
  SCheckSum,SC1,SC2:String;
begin
   for i:=0 to Length(Station)-1 do
   begin

      if  Station[i].Stationcon = 'C' then
      begin
```

```
            CheckSum := Station[i].StationNum + 12;
            SCheckSum := IntToHex(CheckSum,2);
                SC1 := '$'+SCheckSum[1];    CS1 := 15 - strtoint(SC1);
                SC1 := IntToHex(CS1,1);
                SC2 := '$'+SCheckSum[2];    CS2 := 15 - strtoint(SC2);
                SC2 := IntToHex(CS2,1);
                SCheckSum := '$'+SC1+SC2;
                CheckSum := Strtoint(SCheckSum)+1;
                SCheckSum := IntToHex(CheckSum,2);

ApxComPort1.PutString(':'+IntToHex(Station[i].StationNum,2)+'0400000008'+SChec
kSum+#13#10);
            FFinishFlag:=false;
            if CheckTimeOut(TimeOut)then //wait for data
            begin
                 StationTimeOut(Station[i]); //time out
            end else begin
                SetAnalogToStation(Station[i],FBuf);
                //set data to object polled station
            end;
            FBuf:=''; //clear buffer
            CheckSum := Station[i].StationNum + 6;
            SCheckSum := IntToHex(CheckSum,2);
                SC1 := '$'+SCheckSum[1];    CS1 := 15 - strtoint(SC1);
                SC1 := IntToHex(CS1,1);
                SC2 := '$'+SCheckSum[2];    CS2 := 15 - strtoint(SC2);
                SC2 := IntToHex(CS2,1);
                SCheckSum := '$'+SC1+SC2;
                CheckSum := Strtoint(SCheckSum)+1;
                SCheckSum := IntToHex(CheckSum,2);

ApxComPort1.PutString(':'+IntToHex(Station[i].StationNum,2)+'0200000004'+SChec
kSum+#13#10);
            FFinishFlag:=false;
            if CheckTimeOut(TimeOut)then //wait for data
            begin
                 StationTimeOut(Station[i]); //time out
            end else begin
                SetDigitalToStation(Station[i],FBuf);
                //set data to object polled station
            end;
            FBuf:=''; //clear buffer
            CheckSum := Station[i].StationNum + 5;
            SCheckSum := IntToHex(CheckSum,2);
                SC1 := '$'+SCheckSum[1];    CS1 := 15 - strtoint(SC1);
                SC1 := IntToHex(CS1,1);
                SC2 := '$'+SCheckSum[2];    CS2 := 15 - strtoint(SC2);
```

```
            SC2 := IntToHex(CS2,1);
            SCheckSum := '$'+SC1+SC2;
            CheckSum := Strtoint(SCheckSum)+1;
            SCheckSum := IntToHex(CheckSum,2);

ApxComPort1.PutString(':'+IntToHex(Station[i].StationNum,2)+'0100000004'+SChec
kSum+#13#10);
        FFinishFlag:=false;
        if CheckTimeOut(TimeOut)then //wait for data
        begin
            StationTimeOut(Station[i]); //time out
        end else begin
            SetDigitalToStation(Station[i],FBuf);
            //set data to object polled station
        end;
        FBuf:=''; //clear buffer
    end;
   end;
end;

procedure TPolling.StationTimeOut(Sta:TStation);
var //Set time out data to object station
   i:Integer;
begin
     for i:=1 to 8 do
     begin
          Sta.Analog[i]:='T';
     end;
     for i:=1 to 4 do
     begin
          Sta.DigitalInput[i]:='T';
          Sta.DigitalOutput[i]:='T';
     end;
end;

procedure TPolling.SetDigitalToStation(Station:TStation; Data:String);
var
   cmd,dat:String;
   i,Ddat,d1,v:Integer;
   x:extended;
begin
     if Length(Data)=13 then
     begin
          cmd:=Copy(Data,4,5);
          dat:=Copy(Data,8,9);
          cmd := cmd[1]+cmd[2];
          if (cmd = '02')then //check command
          begin
               dat := '$'+dat[1]+dat[2];
               Ddat := strtoint(dat);
```

```
                for i := 1 to 4 do
                begin
                 d1 := Ddat mod 2;
                 Ddat := Ddat div 2;
                 Station.DigitalInput[i] := inttostr(d1);
                end;
            end;
            if (cmd = '01') then //check command
            begin
                 dat := '$'+dat[1]+dat[2];
                 Ddat := strtoint(dat);
                for i := 1 to 4 do
                begin
                 d1 := Ddat mod 2;
                 Ddat := Ddat div 2;
                 Station.DigitalOutput[i] := inttostr(d1);
                end;

            end;
            end else begin
                DataIncorrect(Station);
                end;
end;

procedure TPolling.SetAnalogToStation(Station: TStation; Data: String);
var len, Ddat1: Integer;
   cmd, dat, Ddat: String;
   i, DotCount, CommaCount, k: Integer;
   s, s1: String;
   c: Char;
   err: Boolean;
begin
    len:=Length(Data);
     if len > 4 then
     begin
          cmd:=Copy(Data, 4,5);
          dat:=Copy(Data, 8, len-2);
          cmd := cmd[1]+cmd[2];
          if (cmd = '04') then //check command
          begin
             for i := 0 to 7 do
             begin
             Ddat :='';
               for  k:= 1 to 4 do  Ddat := Ddat+Data[k+7+i*4];
                    Ddat  := '$'+Ddat;
                    Ddat1 := strtoint(Ddat);
                    Ddat := inttostr(Ddat1);
             Station.Analog[i+1] := Ddat;
             end;
```

```
          end;
          end else begin
              DataIncorrect(Station);
              end;
end;

procedure TPolling.DataIncorrect(Sta:TStation);
var //Set time out data to object station
   i:Integer;
begin
     for i:=1 to 8 do
     begin
          Sta.Analog[i]:='X';
     end;
     for i:=1 to 4 do
     begin
          Sta.DigitalInput[i]:='X';
          Sta.DigitalOutput[i]:='X';
    end;
end;


function
TPolling.Ative_Command2(Number_station:Integer;Ch:String;Act:string):Boo
lean;
var CheckSum,CS1,CS2:byte;
   SCheckSum,SC1,SC2:String;
begin
         if Act ='1' then Act:='FF' else Act := '00';
         CheckSum :=Number_station +5+strtoint(Ch)+strtoint('$'+Act);
         SCheckSum := IntToHex(CheckSum,2);
            SC1 := '$'+SCheckSum[1];    CS1 := 15 - strtoint(SC1);
            SC1 := IntToHex(CS1,1);
            SC2 := '$'+SCheckSum[2];    CS2 := 15 - strtoint(SC2);
            SC2 := IntToHex(CS2,1);
            SCheckSum := '$'+SC1+SC2;
           CheckSum := Strtoint(SCheckSum)+1;
           SCheckSum := IntToHex(CheckSum,2);

ApxComPort1.PutString(':'+IntToHex(Number_station,2)+'05000'+Ch+Act+'00'+SCh
eckSum+#13#10); //Send Read Analog Input
        FFinishFlag:=false;
        FActiveCommand2:=true;
        if CheckTimeOut(100)then //Wait for data
        begin
           Result:= not FActiveCommand2;
           FBuf:='';
           //Continue;//poll next station
        end else begin
           Result:= FActiveCommand2;;
```

```
        end;
        FBuf:='';
end;


procedure TPolling.ReadAnalogType(var Station:array of TStation);
var
   i:Integer;
   CheckSum,CS1,CS2:byte;
   SCheckSum,SC1,SC2:String;
begin
   for i:=0 to Length(Station)-1 do
   begin
      if Station[i].Stationcon ='C' then
       begin
        CheckSum := Station[i].StationNum +11;
         SCheckSum := IntToHex(CheckSum,2);
            SC1 := '$'+SCheckSum[1];    CS1 := 15 - strtoint(SC1);
            SC1 := IntToHex(CS1,1);
            SC2 := '$'+SCheckSum[2];    CS2 := 15 - strtoint(SC2);
            SC2 := IntToHex(CS2,1);
            SCheckSum := '$'+SC1+SC2;
            CheckSum := Strtoint(SCheckSum)+1;
            SCheckSum := IntToHex(CheckSum,2);

ApxComPort1.PutString(':'+IntToHex(Station[i].StationNum,2)+'0300000008'+SChec
kSum+#13#10);
        FFinishFlag:=false;
        if CheckTimeOut(Timeout)then //Wait for data
        begin
           StationTimeOut(Station[i]); //time out
           FBuf:='';
           Continue;//poll next station
        end else begin
            SetTypeAnalog(Station[i],FBuf);
//set data to object polled station
//*** FFinishFlag Active =Receive data completed
        end;
        FBuf:='';
      end;
   end;
end;

procedure TPolling.SetTypeAnalog(Station:TStation; Data:String);
var
   len,Ddat1:Integer;
   cmd,dat,Ddat:String;
   i,DotCount,CommaCount,k:Integer;
```

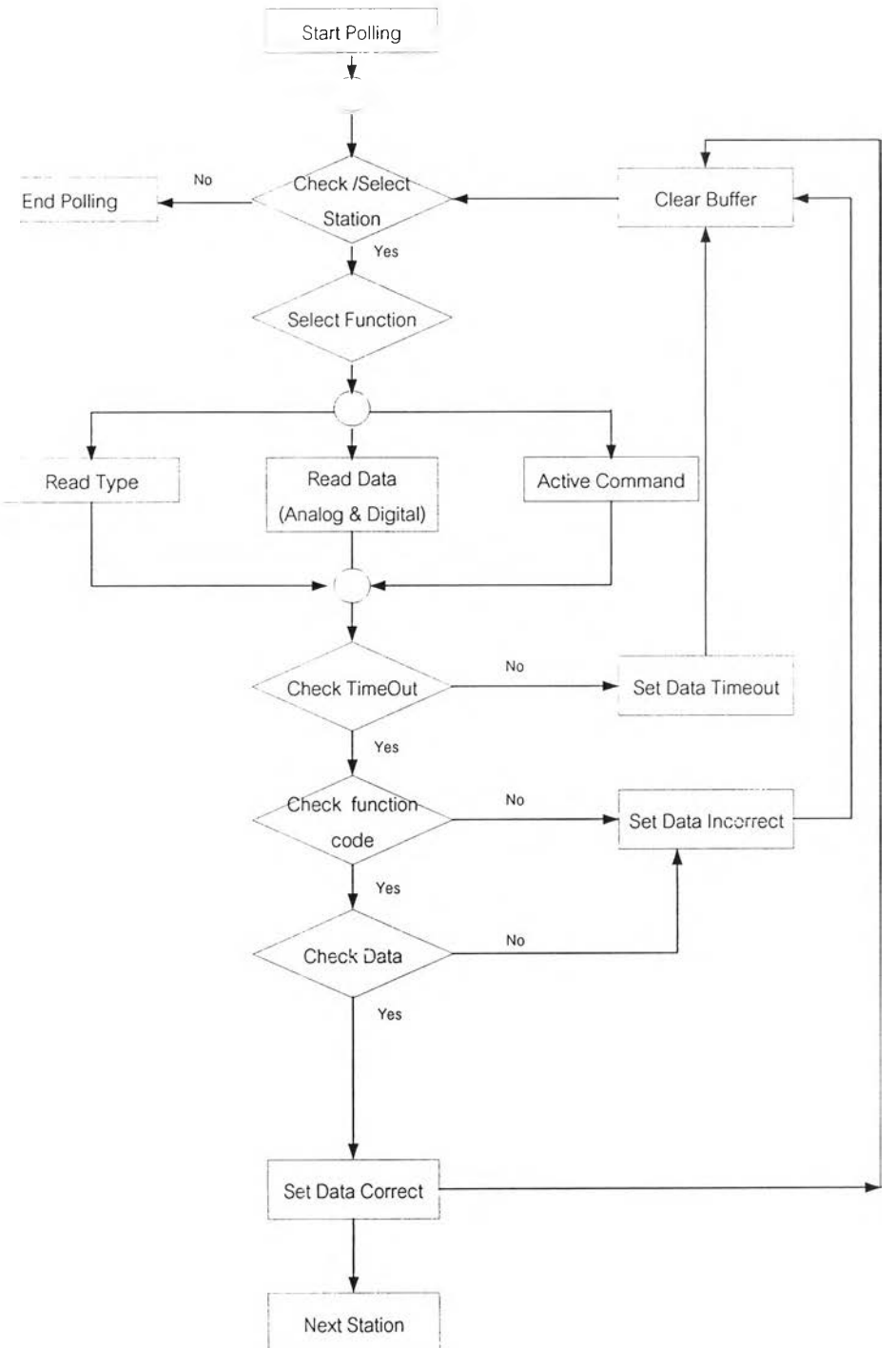
```
    s,s1:String;
    c:Char;
    err:Boolean;
begin
     len:=Length(Data);
      if len > 4 then
      begin
            cmd:=Copy(Data,4,5);
            dat:=Copy(Data,8,len-2);
            cmd := cmd[1]+cmd[2];
            if (cmd = '03') then //check command
            begin
               for i := 0 to 7 do
               begin
               Ddat :='';
                 for  k:= 1 to 4 do  Ddat := Ddat+Data[k+7+i*4];
                     Ddat  := '$'+Ddat;
                     Ddat1 := strtoint(Ddat);
                     Ddat := inttostr(Ddat1);
               Station.AnalogType[i+1]:= Ddat;

     end;

            end;
            end else begin
                DataIncorrect(Station);
                end;
end;


end.
```

3. ตัวอย่างซอฟต์แวร์สำหรับการการอ่านข้อมูลจากสถานีปลายทางระยะไกลโดยการโพลลิ่ง (Polling) ของระบบสกาดาโดยโพรโตคอลเฉพาะของอุปกรณ์ที่นำมาทดสอบที่นำมาทดสอบ



รูปที่ 2 แสดงโครงสร้างการโพลลิ่งข้อมูลโดยใช้โพรโตคอลเฉพาะของอุปกรณ์ที่นำมาทดสอบที่นำมาทดสอบ

```
unit Ppolling;

interface
uses
  SysUtils, Types, Classes, Variants, QGraphics, QControls, QForms,
QDialogs,
  QTypes, AxMisc, AxPort,Station,IdGlobal;

{*****************เพื่อให้ง่ายในตัวอย่างนี้ใช้ Component การสื่อสารแบบอนุกรม (Serial) ของ
  TurboPower Async Professional CLX 1.01 Download ฟรีที่ Sourceforge.net
 *****************}


type
    TPolling =class (TComponent)

    private
      FBuf: String;
      FFinishFlag: Boolean;
      FActiveCommand:Boolean;
      procedure ApxComPort1TriggerAvail(CP: TObject; Count: Word);
      function CheckTimeOut(TimeOut: Cardinal):Boolean;
      procedure DataIn();
      procedure StationTimeOut(Sta: TStation);
      procedure SetAnalogToStation(Station: TStation;Data: String);
      procedure SetDigitalToStation(Station: TStation;Data: String);
      procedure SetTypeAnalog(Station: TStation; Data: String);
      procedure DataIncorrect(Sta: TStation);
    public
      ApxComPort1: TApxComPort;
      TimeOut: Cardinal;
      constructor Create(AOwner: TComponent);override;
      destructor Destroy;override;
      procedure Poll(var Station: Array of TStation);
      Procedure ScanStation(var Station: array of TStation);
      function
Ative_Command(Number_station:integer;A1:string;A2:string;A3:string;A4:str
ing):Boolean;
      procedure ReadAnalogType(var Station: array of TStation);
    end;
implementation
{ TPolling }

constructor TPolling.Create(AOwner: TComponent);
begin
  inherited;
  ApxComPort1:=TApxComPort.Create(self); //Create Asyn Pro Component
  ApxComPort1.OnTriggerAvail:=ApxComPort1TriggerAvail; //Set event
method
  ApxComPort1.ComNumber:=1;  //Set Default Port Com1
  Timeout:=850;  //Set Default Time Out
```

```
end;

destructor TPolling.Destroy;
begin
  ApxComPort1.Open:=False; //Close Comport
  ApxComPort1.Destroy; //Destroy Ayn Pro Component
  inherited;
end;

procedure TPolling.ApxComPort1TriggerAvail(CP: TObject; Count: Word);
var //PpdComPort's OnTriggerAvail Function
   i: Integer;
   c: Char;
begin
   for i:=1 to count do //Read data to buffer
   begin
      c:=ApxComPort1.GetChar;
      FBuf:=FBuf+c;
      if c=#13 then DataIn;  //Finish Data Stream
   end;
end;

function TPolling.CheckTimeOut(TimeOut: Cardinal): Boolean;
var
   StartTime: Cardinal;
   StopTime: Cardinal;
begin
     StartTime:=GetTickCount;
     repeat
           Application.ProcessMessages;
           StopTime:=GetTickCount;
     until ((FFinishFlag)or(StopTime-StartTime>TimeOut));
        //Exit when Timeout or have data
     Result:=not FFinishFlag;
end;

procedure TPolling.DataIn;
begin
   FFinishFlag:=true; //Clear flag when data complete
end;

procedure TPolling.Poll(var Station:array of TStation);
var
   i: Integer;
begin
   for i:=0 to Length(Station)-1 do
   begin
      if Station[i].Stationcon = 'C' then
       begin

ApxComPort1.PutString('#'+IntToHex(Station[i].StationNum,2)+'RAIF'+#13);
```

```
//Send Read Analog Input
        FFinishFlag:=false;
        if CheckTimeOut(TimeOut)then //Wait for data
        begin
           StationTimeOut(Station[i]); //time out
           FBuf:='';
           Continue;//poll next station
        end else begin
            SetAnalogToStation(Station[i],FBuf); //set data to object
polled station   *** FFinishFlag Active =Receive data completed
        end;
        FBuf:='';

ApxComPort1.PutString('#'+IntToHex(Station[i].StationNum,2)+'RDIO'+#13);
//Send Read Didital IO
        FFinishFlag:=false;
        if CheckTimeOut(TimeOut)then //wait for data
        begin
             StationTimeOut(Station[i]); //time out
     end else begin
            SetDigitalToStation(Station[i],FBuf); //set data to object
polled station
        end;
        FBuf:=''; //clear buffer
      end;
   end;
end;

procedure TPolling.StationTimeOut(Sta:TStation);
var //Set time out data to object station
   i:Integer;
begin
     for i:=1 to 8 do
     begin
          Sta.Analog[i]:='T';
          sta.AnalogType[i]:='T';
     end;
     for i:=1 to 4 do
     begin
          Sta.DigitalInput[i]:='T';
          Sta.DigitalOutput[i]:='T';
     end;
end;

procedure TPolling.SetDigitalToStation(Station:TStation; Data:String);
var
   cmd,dat:String;
   i:Integer;
begin
     if Length(Data)=13 then
```

```
    begin
        cmd:=Copy(Data,1,4);
        dat:=Copy(Data,5,8);
        if (cmd = 'DIO>')then //check command
        begin
            for i:=1 to 4 do
            begin
                if dat[i]in ['0','1']then //check data
                begin
                    Station.DigitalInput[i]:=dat[i];
                end else begin
                    Station.DigitalInput[i]:='X'; //data incorrect
                end;
            end;
            for i:=1 to 4 do
            begin
                if dat[4+i]in ['0','1']then //check data
                begin
                    Station.DigitalOutput[i]:=dat[4+i];
                end else begin
                    Station.DigitalOutput[i]:='X'; //data incorrect
                end;
            end;
        end else begin
            DataIncorrect(Station);
        end;
    end;
end;

procedure TPolling.SetAnalogToStation(Station: TStation; Data: String);
var
   len: Integer;
   cmd,dat: String;
   i,DotCount,CommaCount: Integer;
   s: String;
   c: Char;
   err: Boolean;
begin
    len:=Length(Data);
    if len>4 then
    begin
        cmd:=Copy(Data,1,3);
        dat:=Copy(Data,4,len-4);
        dat:=dat+',';
        if cmd='AI>' then //check command
        begin
            Commacount:=0;
            DotCount:=0;
            s:='';
            err:=false;
```

```
                for i:=1 to Length(dat)do
                begin

                    c:=dat[i];
                    if c in ['0'..'9','.']then
                    //check correct real number format
                    begin
                        s:=s+c;
                        if c='.' then inc(DotCount);
                    end else if c ='-' then
                    begin
                        s:=s+c;
                    end
                    else if c=',' then //finish one unit real data
                    begin
                        inc(CommaCount);
                           if ((DotCount>1)or(err))then
                           begin
                               Station.Analog[CommaCount]:='X';
                               if (err)and(s='OVR')then
                               Station.Analog[CommaCount]:='O';
                               //Data over range
                               if (err)and(s='N')then
                               Station.Analog[CommaCount]:='N';
                               //Analog not connect
                               if (err)and(s='UNR')then
                               Station.Analog[CommaCount]:='U';
                               //Data under range
                               end else begin
                               Station.Analog[CommaCount]:=s;
                           end;
                        DotCount:=0;
                        s:='';
                        err:=false;
                    end else begin
                        err:=true;
                        s:=s+c;
                    end;
                end;
            end else begin
                DataIncorrect(Station);
            end;
        end;
end;

procedure TPolling.DataIncorrect(Sta:TStation);
var //Set time out data to object station
   i:Integer;
begin
```

```
      for i:=1 to 8 do
      begin
           Sta.Analog[i]:='X';
           sta.AnalogType[i]:='X';
      end;
      for i:=1 to 4 do
      begin
           Sta.DigitalInput[i]:='X';
           Sta.DigitalOutput[i]:='X';
     end;
end;

Procedure TPolling.ScanStation(var Station:array of TStation);
var i:Integer;
begin
 for i:=0 to Length(Station)-1 do
   begin

ApxComPort1.PutString('#'+IntToHex(Station[i].StationNum,2)+'x'+#13); //Send
Read Analog Input
        FFinishFlag:=false;
        if CheckTimeOut(100)then //Wait for data
        begin
           Station[i].Stationcon := 'NC';
           FBuf:='';
           Continue;//poll next station
        end else begin
           Station[i].Stationcon := 'C';
        end;
       FBuf:='';
    end;
 //Ative_Command(Station,A1,A2,A3,A4);
end;

 function
TPolling.Ative_Command(Number_station:Integer;A1:string;A2:string;A3:str
ing;A4:string):Boolean;
begin

ApxComPort1.PutString('#'+IntToHex(Number_station,2)+'WDO1234,'+A1+A2+A3+A4+
#13); //Send Read Analog Input
        FFinishFlag:=false;
        FActiveCommand:=true;
       if CheckTimeOut(100)then //Wait for data
        begin
           Result:= not FActiveCommand;
           FBuf:='';
           Continue;//poll next station
        end else begin
           Result:= FActiveCommand;;
```

```
        end;
        FBuf:='';
end;

procedure TPolling.ReadAnalogType(var Station:array of TStation);
var
   i:Integer;
begin
   for i:=0 to Length(Station)-1 do
   begin
      if Station[i].Stationcon ='C' then
       begin

ApxComPort1.PutString('#'+IntToHex(Station[i].StationNum,2)+'RTY'+#13);
//Send Read Analog Input
        FFinishFlag:=false;
        if CheckTimeOut(Timeout) then //Wait for data
        begin
           StationTimeOut(Station[i]); //time out
           FBuf:='';
           Continue;//poll next station
        end else begin
            SetTypeAnalog(Station[i],FBuf); //set data to object polled
station   *** FFinishFlag Active =Receive data completed
        end;
        FBuf:='';
      end;
   end;
end;

procedure TPolling.SetTypeAnalog(Station:TStation; Data:String);
var
   len:Integer;
   cmd,dat: String;
   i,DotCount,CommaCount:Integer;
   s:String;
   c:Char;
   err:Boolean;
begin
     len:=Length(Data);
     if len>4 then
     begin
          cmd:=Copy(Data,1,5);
          dat:=Copy(Data,6,len-6);
          dat:=dat+',';
          if cmd='TYPE>' then //check command
          begin
               Commacount:=0;
               DotCount:=0;
               s:='';
```

```
                err:=false;
                for i:=1 to Length(dat)do
                begin
                     c:=dat[i];
                     if c in ['0'..'9','.']then
                      //check correct real number format
                     begin
                          s:=s+c;
                          if c='.' then inc(DotCount);
                     end else if c=',' then //finish one unit real data
                      begin
                          inc(CommaCount);
                             if ((DotCount>1)or(err))then
                             begin
                                  Station.AnalogType[CommaCount]:='X';
                                 if (err)and(s='OVR')then
                                  Station.AnalogType[CommaCount]:='O';
                                  //Data over range
                                  if (err)and(s='N')then
                                  Station.AnalogType[CommaCount]:='N';
                                  //Analog not connect
                                  if (err)and(s='UNR')then
                                  Station.AnalogType[CommaCount]:='U';
                                  //Data under range
                             end else begin
                                  Station.AnalogType[CommaCount]:=s;
                             end;
                          DotCount:=0;
                          s:='';
                          err:=false;
                     end else begin
                          err:=true;
                          s:=s+c;
                     end;
                end;
           end else begin
               DataIncorrect(Station);
           end;
     end;
end;

end.
```

4. ตัวอย่างการสร้างหน่วยการเก็บข้อมูลสำหรับการโพลลิ่งข้อมูล (Polling Data)

```
unit Station;

interface
type
  Tstation =class
public
  StationNum : Byte;
  Analog : Array[1..8]of String;
  AnalogType : Array[1..8]of String;
  DigitalInput : Array[1..4]of String;
  DigitalOutput : Array[1..4]of String;
  Stationcon : string;

constructor Create;
end;

implementation
{Tstation}
constructor TStation.Create;
var
  i:integer;
begin
    for i:=1 to 8 do
     begin
        Analog[i]:='N';
        AnalogType[i]:='N';
     end;
     for i:=1 to 4 do
    begin
    DigitalInput[i]:='N';
    end;
    for i:=1 to 4 do
    begin
    DigitalOutput[i]:='N';
    end;

  end;
end.
```

ภาคผนวก ข
รายละเอียดการติดตั้งและกำหนดค่าเริ่มต้นก่อนการใช้งานระบบควบคุม และรวบรวมข้อมูลสกาดาบนระบบปฏิบัติการลีนุกซ์

1. **การเตรียมความพร้อมสำหรับระบบปฏิบัติการเพื่อการใช้งานซอฟต์แวร์สกาดาต้นแบบ**

สำหรับการใช้งานซอฟต์แวร์สกาดาต้นแบบมีการแบ่งการทำงานเป็นสองส่วนคือ ส่วนที่ทำหน้าที่เป็นซิร์ฟเวอร์จะเรียกการใช้งานผ่านโปรแกรมที่ชื่อว่า MainServer และส่วนที่สองจะทำหน้าที่เป็นไคลเอนท์มีการเรียกใช้งานผ่านโปรแกรมที่ชื่อว่า Monitor ต้องมีการตรวจสอบ เตรียมความพร้อมและกำหนดคุณสมบัติให้กับระบบปฏิบัติการดังต่อไปนี้

1.1 ระบบปฏิบัติการที่ทำหน้าที่เป็นเซิร์ฟเวอร์ ถูกสร้างขึ้นบนระบบปฏิบัติการลีนุกซ์มีรายละเอียดข้อกำหนดดังต่อไปนี้

- ข้อกำหนดทางฮาร์ดแวร์
  - หน่วยประมวลผลกลาง (CPU) เทียบเท่ากับ Pentium II 400 เมกกะเฮิรตซ์ หรือมากกว่า
  - แรม (Ram) อย่างน้อย 256 เมกกะไบต์
  - การ์ดแสดงผล (VGA or Monitor Card) ซึ่งมีแรมอย่างน้อย 16 เมกกะไบต์
  - ฮาร์ดดิสก์ (Hard Disk) 20 กิกกะไบต์ หรือมากกว่า
- ข้อกำหนดของระบบปฏิบัติการ
  - ระบบปฏิบัติการลีนุกซ์ของบริษัทเรดแฮ็ท รุ่น 7.2 – 8 (RedHat 7.2 – 8)
  - ติดตั้งการใช้งานการให้บริการฐานข้อมูล
  - ติดตั้งการใช้งานภาษาไทยโดยใช้รูปแบบของ TIS620 [15]
- ข้อกำหนดการใช้ฐานข้อมูล
  - ติดตั้งตัวให้บริการฐานข้อมูลมายเอสคิวแอล (MySQL Database Server) ในวิทยานิพนธ์ฉบับนี้ใช้ มายเอสคิวแอล รุ่น 3.23.41 ซึ่งแถมมากับแผ่นการติดตั้งระบบปฏิบัติการลีนุกซ์ RedHat 7.2 และสามารถใช้กับมายเอสคิวแอลรุ่นที่สูงกว่าได้ด้วย โดยขั้นตอนการติดตั้งผู้ติดตั้งจะต้องลงโปรแกรมทั้งในส่วนของไคลเอนท์และเซิร์ฟเวอร์ของตัวบริการฐานข้อมูลมายเอสคิวแอลให้ครบถ้วนหลังจากนั้นให้นำไฟล์ของฐานข้อมูลที่ชื่อ ADdata (อยู่ใน CD ที่แนบมาด้วย) มาบันทึกลงไปในฐานข้อมูลมายเอสคิวแอลที่ติดตั้งใหม่ซึ่งจะได้ฐานข้อมูล

ชื่อ ADdata และตารางสำหรับบันทึกข้อมูลประเภทต่างๆ สำหรับการแสดงผลขณะเวลาจริงและสำหรับเก็บฐานข้อมูลประวัติรวมทั้งหมด 7 ตาราง

- ในกรณีที่ต้องการสร้างฐานข้อมูลหรือตารางข้อมูลใหม่จะต้องมีการสร้างฐานข้อมูลไว้ใช้สำหรับการแสดงผลขณะเวลาจริงและสำหรับเก็บฐานข้อมูลประวัติ ตามรายละเอียดการออกแบบฐานข้อมูลในบทที่ 3 และตัวอย่างโปรแกรมทดสอบการทำงานกับฐานข้อมูลในตารางต่างๆในส่วนของโปรแกรม MainServer ส่วนของ Main.pas ซึ่งมีการระบุไว้อย่างชัดเจนถึงการติดต่อฐานข้อมูลชื่ออะไร และตารางไหนซึ่งก่อนจะนำซอฟต์แวร์ต้นแบบมาใช้ต้องทำตรวจสอบฐานข้อมูลและตารางข้อมูลก่อนทุกครั้งแต่ถ้าต้องการสร้างใหม่ให้อ่านจากลักษณะโครงสร้างที่ออกแบบไว้และผลการทดสอบตามบทที่ 3 และ 4 อย่างเช่นต้องการสร้างตารางของฐานข้อมูลไว้สำหรับเก็บข้อมูลชนิดของอุปกรณ์ต้องมีการกำหนดดังต่อไปนี้

ตัวอย่างการสร้างตารางข้อมูล

สมมติว่าต้องการสร้างตารางของฐานข้อมูลไว้สำหรับเก็บข้อมูลชนิดของอุปกรณ์แสดงรายละเอียดได้ดังตารางที่ 1

เมื่อกำหนดคุณสมบัติของตารางเรียบร้อยแล้วให้ทดลองพิมพ์คำสั่งต่อไปนี้ที่ Shell

[root@localhost root]# mysql // เรียกการใช้ฐานข้อมูลมายเอสคิวแอล

[root@localhost root]# Create Database ADdata; // สร้างฐานข้อมูลชื่อ ADdata

[root@localhost root]# Show Databases; //ตรวจสอบฐานข้อมูลที่สร้างขึ้น

[root@localhost root]# Use ADdata; // เรียกใช้ฐานข้อมูลที่สร้างขึ้น

[root@localhost root]# Create Table SetType (Station_Type int(7),Datetimes Datetime,AT1 Char(2) ,AT2 Char(2),AT3 Char(2) ,AT4 Char(2) ,AT5 Char(2), AT6 Char(2) ,AT7 Char(2) ,AT8 Char(2), COF1 Char(3) ,COF2 Char(3) , COF3 Char(3) , COF4 Char(3) , COF5 Char(3) , COF6 Char(3) , COF7 Char(3), COF8 Char(3)); // สร้างตารางที่ออกแบบไว้ (ตัวเลขที่กำหนดเป็นค่าที่สมมติขึ้น)

[root@localhost root]# Show Tables; // ตรวจสอบตารางที่สร้างขึ้น

[root@localhost root]# Describe SetType; // ตรวจสอบชนิดข้อมูลในตารางที่สร้างขึ้น

ตารางที่ 1 การสร้างตารางสำหรับเก็บข้อมูลชนิดของอุปกรณ์

Data Name	Name Field	Data Type	Maximum Memory
Name Station	Station_Type	Unsigned Tiny Integer	1 byte
Datetime	Datetime	Datetime	8 byte
Analog Type 1	AT1	Character	2 byte
Analog Type 2	AT2	Character	2 byte
Analog Type 3	AT3	Character	2 byte
Analog Type 4	AT4	Character	2 byte
Analog Type 5	AT5	Character	2 byte
Analog Type 6	AT6	Character	2 byte
Analog Type 7	AT 7	Character	2 byte
Analog Type 8	AT 8	Character	2 byte
Coefficient Of AT1	COF 1	Character	3 byte
Coefficient Of AT2	COF 2	Character	3 byte
Coefficient Of AT3	COF 3	Character	3 byte
Coefficient Of AT4	COF 4	Character	3 byte
Coefficient Of AT5	COF 5	Character	3 byte
Coefficient Of AT6	COF 6	Character	3 byte
Coefficient Of AT7	COF 7	Character	3 byte
Coefficient Of AT8	COF 8	Character	3 byte
		Total	49 byte

<u>ตัวอย่างโปรแกรมการติดต่อกับฐานข้อมูล</u>

```
dbstationtype.Host := 'localhost';
        // เป็นการติดต่อเครื่อง Database Server จากเครื่องเซิร์ฟเวอร์เอง
dbstationtype.Database := 'ADdata';
        // เป็นการติดต่อฐานข้อมูลชื่อ ADdata ซึ่งเป็นแหล่งเก็บตารางข้อมูลสำหรับการ
           แสดงผลขณะเวลาจริงและสำหรับเก็บฐานข้อมูลประวัติ ตามรายละเอียดการ
           ออกแบบฐานข้อมูลในบทที่ 3 และการทดสอบในบทที่ 4
dbstationtype.Login := 'root';
dbstationtype.Password := 'root ';
tbstationtype.TableName :=  'SetType';
        // เป็นการติดต่อกับตารางฐานข้อมูลชื่อ SetType ซึ่งมีหน้าที่ในการเก็บประเภท
           ของสัญญาณแอนะล็อกของแต่ละช่องสัญญาณที่ต้องการวัด
tbstationtype.Open;             // เริ่มการใช้ฐานข้อมูล
```

- ข้อกำหนดการติดต่อสื่อสารผ่านทางพอร์ตอนุกรม การเรียกใช้งานผ่านพอร์ตอนุกรมจะถูกเรียกใช้ผ่านโปรแกรม MainServer โดยรูปแบบการเก็บข้อมูลจะถูกกำหนดไว้ในส่วนการ Polling ซึ่งถ้าต้องการที่จะเปลี่ยนการใช้งานเป็น RTU ชนิดอื่นต้องทำการกำหนดส่วนการ Polling ใหม่ (แสดงตัวอย่างการกำหนดไว้ที่ ภาคผนวก ก) การเรียกโปรแกรม MainServer ขึ้นมาใช้งานนั้นจะต้องเรียกใช้โดย Root เท่านั้นแต่ถ้า User เป็นผู้เรียกใช้ต้องทำการ Su Root และ Chmod การทำงานของพอร์ตอนุกรมที่เรียกใช้ก่อนการใช้งานทุกครั้ง
- ข้อกำหนดการติดต่อสื่อสารบน TCP/IP เนื่องจากได้มีการออกแบบการใช้งานของซอฟต์แวร์สกาดาเป็นแบบไคลเอนท์/เซิร์ฟเวอร์จึงต้องมีการกำหนดดังต่อไปนี้
  - ต้องกำหนดหมายเลขเครื่องที่ทำหน้าที่เป็นเซิร์ฟเวอร์ (IP - Address)
  - ต้องกำหนดการทำงานของเครื่องเซิร์ฟเวอร์ให้ทำหน้าที่เป็นเครื่องที่ให้บริการฐานข้อมูล

1.2 ระบบปฏิบัติการที่ทำหน้าที่เป็นไคลเอนท์ มีการกำหนดคุณสมบัติที่ใช้กับโปรแกรมชื่อ Monitor ดังต่อไปนี้

- ใช้ได้กับระบบปฏิบัติการไมโครซอฟต์วินโดว์ 2000 โดยเรียกโปรแกรมชื่อ Monitor.exe และต้องมีการลงส่วนของโปรแกรมที่ชื่อ libmysql.dll และ qtintf.dll ไว้ที่ Directory ~\System32 เพื่อช่วยในการทำงานแบบไคลเอนท์/เซิร์ฟเวอร์
- ใช้ได้กับระบบปฏิบัติการลีนุกซ์โดยมีการเรียกผ่านการทำงานโปรแกรมชื่อ Monitor และต้องมีการลงส่วนของโปรแกรมที่ชื่อ libmysqlclient.so ไว้ที่ User/lib เพื่อช่วยในการทำงานแบบไคลเอนท์/เซิร์ฟเวอร์

2. การติดตั้งซอฟต์แวร์สกาดาต้นแบบ

ในส่วนนี้จะแบ่งการติดตั้งในสองส่วนคือในส่วนของเซิร์ฟเวอร์และไคลเอนท์ โดยมีการติดตั้งดังนี้

เซิร์ฟเวอร์

Login Linux โดย Root

นำไฟล์การใช้งานที่ชื่อ MainServer มาใส่ไว้ที่ Directory /home/....

ทดสอบการใช้งานโดยเรียก ~/home/.../ ./MainServer จาก Shell

ถ้าไม่สามารถใช้งานได้ให้กำหนดการเรียกใช้โปรแกรมใหม่ที่ภาคผนวก ข หัวข้อที่ 4 และ 5

ไคลเอนท์

Login Linux โดย Root หรือ User

นำการใช้งานไฟล์ที่ชื่อ Monitor มาใส่ไว้ที่ Directory /User/....

ทดสอบการใช้งานโดยเรียก ~/User/.../ ./Monitor จาก Shell

ถ้าไม่สามารถใช้งานได้ให้กำหนดการเรียกใช้โปรแกรมใหม่ที่ภาคผนวก ข หัวข้อที่ 4 และ 5

ส่วนการใช้งานบนระบบปฏิบัติการไมโครซอฟต์วินโดว์นำไฟล์ที่ชื่อ Monitor.exe มาใส่ไว้ที่ Desktop ทดสอบการใช้งานโดย Double Click ที่ Monitor.exe ถ้าไม่สามารถใช้งานได้ให้กำหนดตามหัวข้อที่ 1.2 อีกครั้ง

3. **การกำหนดให้สิทธิผู้ใช้งานกับระบบฐานข้อมูลมายเอสคิวแอล** (MySql)

จากระบบระบบควบคุมและรวบรวมข้อมูลสกาดาบนระบบปฏิบัติการลีนุกซ์ถูกออกแบบมาให้ใช้มีการใช้งานในรูปแบบของไคลเอนท์/เซิร์ฟเวอร์ดังนั้นเมื่อติดตั้งการใช้งานของระบบจะต้องมีการกำหนดสิทธิของผู้ใช้บริการในส่วนของไคลเอนท์จากส่วนของระบบเซิร์ฟเวอร์โดยรูปแบบในการกำหนดดังต่อไปนี้

- **การเพิ่มผู้ใช้ใหม่** คำสั่งที่ใช้ในการเพิ่มผู้ใช้ของฐานข้อมูลมายเอสคิวแอล คือ INSERT โดยเป็นการเพิ่มผู้ใช้ใน ตารางผู้ใช้ภายใต้ฐานข้อมูลมายเอสคิวแอล มีตัวอย่างการเพิ่มผู้ใช้ดังต่อไปนี้

Mysql> Insert into user values ('161.200.86.19', 'Green', password('hope') ,'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y');

คำอธิบาย

เป็นการเพิ่มผู้ใช้ใหม่สำหรับระบบฐานข้อมูล โดยผู้ใช้ชื่อ Green รหัสผ่านคือ hope ซึ่ง IP ของเครื่องลูกข่ายที่ติดต่ออยู่ คือ161.200.86.19 และกำหนดให้สามารถใช้สิทธิ์ได้ทั้งหมด ('Y') เช่นการอนุญาตให้ผู้ใช้ อ่าน,ลบ หรือแก้ไขได้ เป็นต้นซึ่งรายละเอียดของสิทธิ์ต่างจะขึ้นกับเวอร์ชันของมายเอสคิวแอล

- **การอนุญาตในการใช้ฐานข้อมูล** เมื่อเพิ่มผู้ใช้เรียบร้อยแล้วผู้ใช้จะยังไม่สามารถใช้งานได้จนกว่าจะมีการอนุญาตให้ผู้ใช้นั้นสามารถทำงานกับฐานข้อมูลได้โดยใช้คำสั่งดังต่อไปนี้

Mysql> grant all on test.* to Green;

คำอธิบาย

เป็นการกำหนดให้ผู้ใช้ที่ชื่อ Green ให้สามารถทำได้ทุกอย่างกับฐานข้อมูลชื่อ Test

4. **การเรียกใช้งานซอฟต์แวร์สกาดาบนระบบปฏิบัติการลีนุกซ์สำหรับเครื่องคอมพิวเตอร์ที่มีโปรแกรม Kylix 3 ติดตั้งอยู่**

สามารถเรียกโดยใช้คำสั่งต่อไปนี้

Console 1

host#source /usr/local/kylix3/<bin or lib>/kylixpath

host# ./Server // เรียกการทำงานของส่วนระบบให้บริการ

Console 2

host#source /usr/local/kylix3/<bin or lib>/kylixpath

host# ./Monitor // เรียกการทำงานของส่วนลูกข่าย

5. **การเรียกใช้งานซอฟต์แวร์สกาดาบนระบบปฏิบัติการลีนุกซ์สำหรับเครื่องคอมพิวเตอร์ที่ไม่มีโปรแกรม** Kylix 3 **ติดตั้งอยู่**

ในขั้นตอนนี้สามารถทำได้ 2 วิธีคือ

- เรียกใช้ด้วยวิธีสร้าง file.bash เช่น ./Monitor.bash

ในขั้นแรกต้องทำการหาไฟล์ libborqt-6.9-qt2.3.so , libqt.so.2 มาลงที่เครื่องที่ต้องการใช้งานแล้วทำตามขั้นตอนต่อไปนี้

#!/bin/bash

export LD_LIBRARY_PATH=~/usepath/ pathที่อยู่ของ lib ที่นำมาลงไว้

~/userpath/ Monitor

เวลา Run ก็พิมพ์ ./Monitor.bash

ในส่วนของโปรแกรม Server ก็ทำเช่นเดียวกัน

- บันทึกไฟล์ของ libborqt*.so ทั้งหมดให้ไปเก็บไว้ที่ /usr/lib และเรียกการใช้งานโดยใช้คำสั่ง ./Monitor สำหรับโปรแกรมสำหรับการตรวจสอบสถานะและ ./Server สำหรับการทำงานของสถานีหลัก

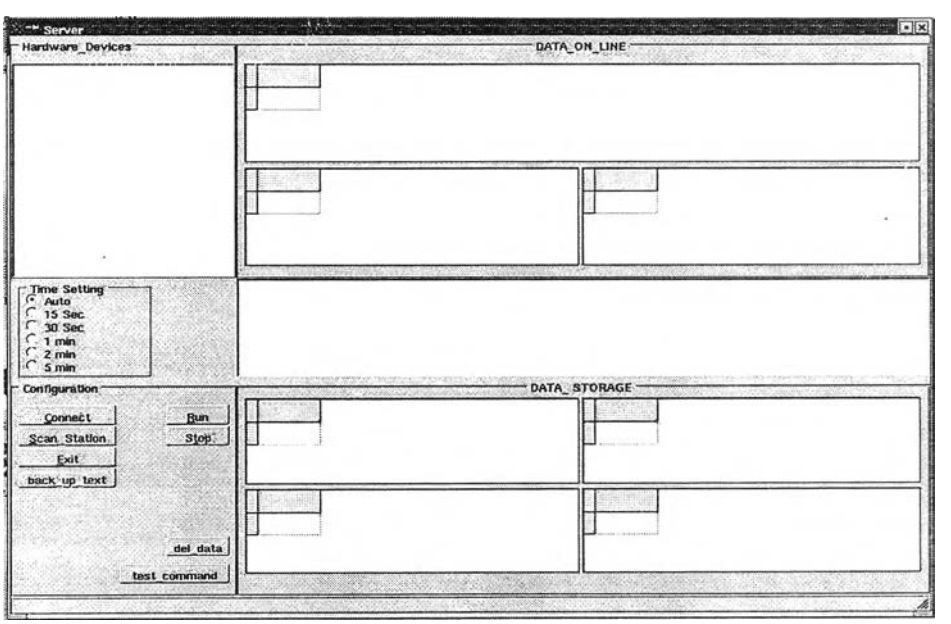
ภาคผนวก ค
คู่มือการใช้ซอฟต์แวร์สกาดา

คู่มือการใช้ซอฟต์แวร์สกาดา

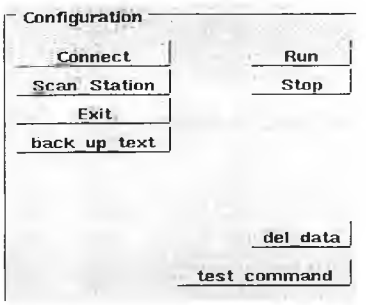
ในภาคผนวก ข ได้กล่าวถึงการเตรียมความพร้อมและการติดตั้งการใช้งานซอฟต์แวร์สกาดาต้นแบบให้กับระบบปฏิบัติการที่ทำหน้าที่เป็นเซิร์ฟเวอร์และไคล์เอนท์และในบทนี้จะกล่าวถึงการใช้งานซอฟต์แวร์สกาดาต้นแบบซึ่งประกอบไปด้วยสองส่วนการทำงานคือส่วนประสานงานผู้ใช้ของระบบให้บริการข้อมูลหรือสถานีหลักโดยโปรแกรมชื่อว่า MainServer ซึ่งมีหน้าที่ในการอ่านข้อมูลจากสถานีปลายทางระยะไกลมาบันทึกลงในฐานข้อมูลเพื่อที่จะช่วยในการแสดงผลเทียบกับเวลาจริงและเป็นฐานข้อมูลประวัติ ส่วนที่สองคือส่วนประสานงานผู้ใช้ของระบบรับบริการข้อมูลหรือมีชื่อเรียกว่าโปรแกรม Monitor มีหน้าที่ในการดึงข้อมูลจากฐานข้อมูลมาแสดงผลในรูปแบบเทียบกับเวลาจริงและรูปแบบของข้อมูลประวัติ โดยในส่วนการทำงานของส่วนประสานงานผู้ใช้ของระบบรับบริการข้อมูลเป็นโปรแกรมที่สามารถใช้งานได้ทั้งระบบปฏิบัติการลีนุกซ์และไมโครซอฟต์วินโดว์

1. การใช้งานในส่วนประสานงานผู้ใช้ของระบบให้บริการข้อมูลหรือสถานีหลัก (การใช้งานโปรแกรม MainServer (Server User Interface Program))

ส่วนประสานงานผู้ใช้ของระบบให้บริการข้อมูลมีหน้าที่ในการตรวจสอบการติดตั้งและกำหนดระยะเวลาในการอ่านข้อมูลของสถานีปลายทางระยะไกลซึ่งมีลักษณะการใช้งานดังต่อไปนี้



รูปที่ 1 หน้าต่างแสดงการทำงานของโปรแกรม MainServer



รูปที่ 2 หน้าต่างส่วนควบคุมการใช้งานของโปรแกรม MainServer

Configuration

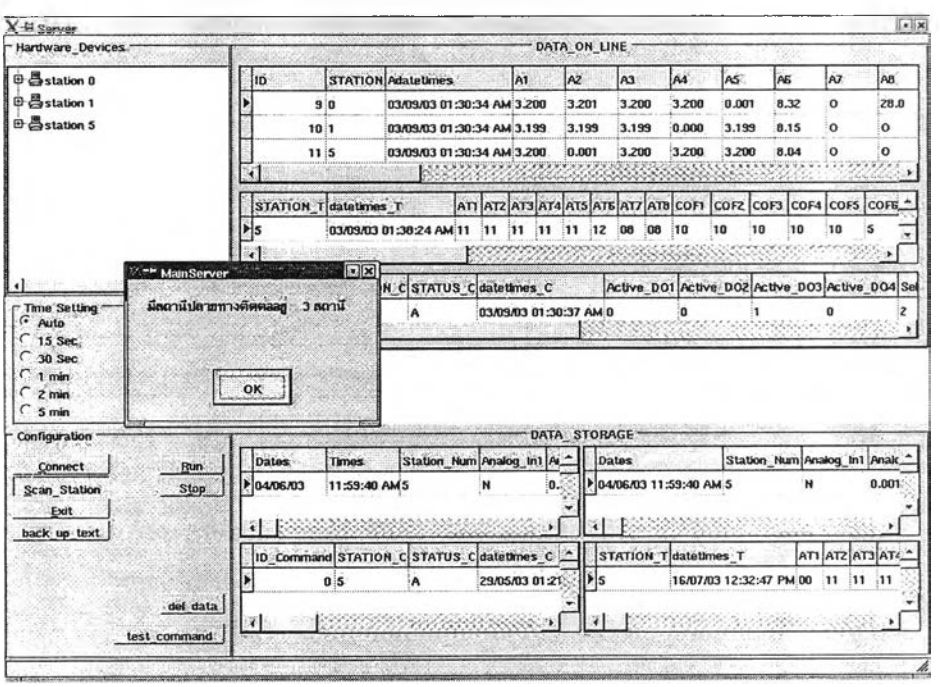
เมื่อแรกเข้าโปรแกรมผู้ใช้ต้องเข้ามาในส่วนนี้เป็นส่วนแรกเพื่อที่จะกำหนดการทำงานให้ระบบโดยผ่านฟังก์ชันการทำงานตามลำดับขั้นดังต่อไปนี้

Connect

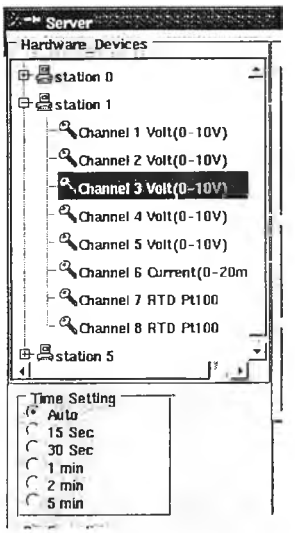
เป็นฟังก์ชันการตรวจสอบการถึงความพร้อมของฐานข้อมูลชนิดต่างๆ (Database) ที่มีการเรียกใช้ในโปรแกรม MainServer

Scan Station

เป็นฟังก์ชันการตรวจสอบจำนวน RTU ที่ติดต่ออยู่กับสถานีหลักรวมถึงการตรวจสอบชนิดของสัญญาณแอนะล็อกของแต่ละสถานีอีกด้วย แสดงดังรูปที่ 3, 4



ดังรูปที่ 3 การตรวจสอบจำนวน RTU



ดังรูปที่ 4 การตรวจสอบชนิดของสัญญาณแอนะล็อกของ RTU และการตั้งเวลาในการอ่านข้อมูลจาก RTU ขึ้นมาเพื่อนำมาบันทึกลงฐานข้อมูล

Time Setting

เป็นฟังก์ชันการตั้งเวลาในการอ่านข้อมูลจาก RTU ขึ้นมาเก็บในฐานข้อมูลเมื่อได้ทำการตรวจสอบแล้วว่าชนิดของสัญญาณแอนะล็อกของ RTU ถูกต้องและพร้อมที่จะทำงาน โดยลักษณะการตั้งเวลามีให้เลือก 2 แบบคือแบบอัตโนมัติ (AUTO) ซึ่งจะทำการตั้งเวลาอ่านค่าจาก RTU ให้เองโดยใช้เวลาอ่านค่าแต่ละ RTU เท่ากับ 2 วินาที ส่วนอีกแบบหนึ่งคือมีให้เลือกอยู่ 5 ช่วงการอ่านข้อมูลจาก RTU คือ 15, 30 วินาที และ 1, 2, 3 นาที

Run / Stop

เป็นฟังก์ชันการควบคุมในการสั่งให้ระบบเริ่มทำงานหรือหยุดทำงาน

Del Data

เป็นฟังก์ชันที่ใช้ในการลบข้อมูลของระบบฐานข้อมูลที่ได้บันทึกไว้ใน Hard Disk ทั้งหมด

Test Command

เป็นฟังก์ชันการตรวจความปกติของการควบคุมจากสถานีปลายทางระยะไกลผ่านสถานีหลักโดยการสุ่มสถานีและช่องสัญญาณขึ้นมาใช้ในการทดสอบ

Back Up to Text

เป็นฟังก์ชันการสำรองข้อมูลทั้งระบบในรูปของ Text file เพื่อความสะดวกในการการเคลื่อนย้ายข้อมูลและนำกลับมาใช้ในรูปฐานข้อมูลอีกก็ได้

Exit

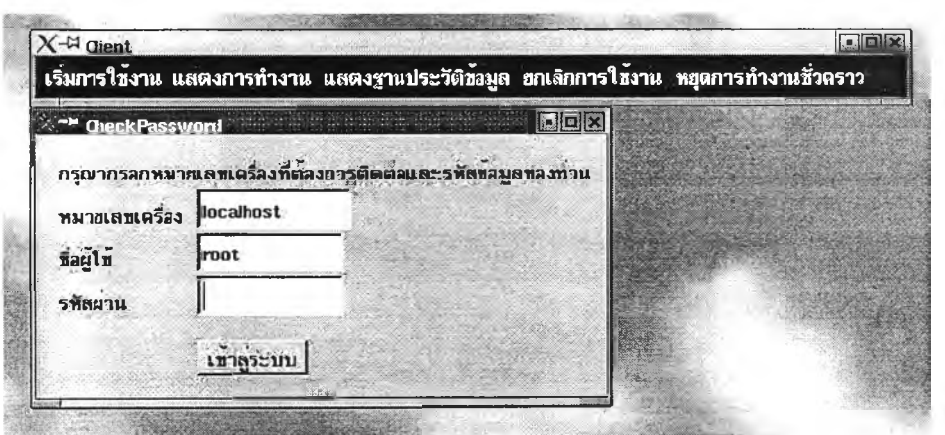
เป็นฟังก์ชันใช้ในการออกจากระบบการทำงานของระบบให้บริการ

2. **การใช้งานในส่วนประสานงานผู้ใช้ของระบบรับบริการข้อมูล** (การใช้งานโปรแกรม Monitor (Client User Interface Program))

โปรแกรม Monitor มีหน้าที่ในการดึงข้อมูลจากฐานข้อมูลมาแสดงผลในรูปแบบเทียบกับเวลาจริงและรูปแบบของข้อมูลประวัติ โดยโปรแกรม Monitor เป็นโปรแกรมที่สามารถใช้งานได้ทั้งระบบปฏิบัติการลีนุกซ์และไมโครซอฟต์วินโดว์ มีลำดับขั้นการทำงานดังต่อไปนี้

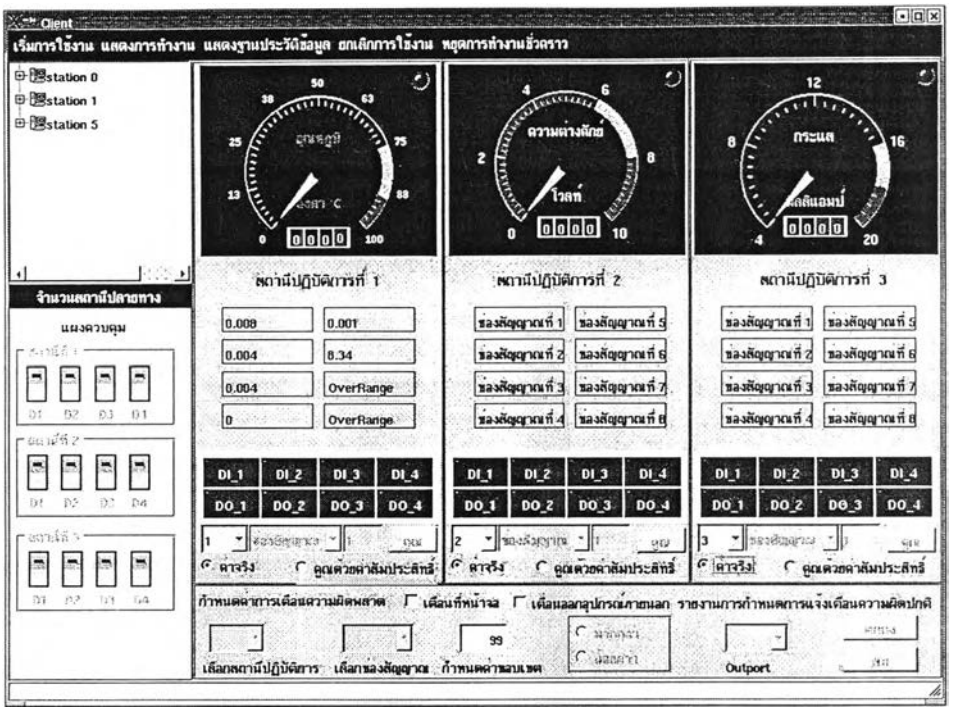
เริ่มการใช้งาน

เป็นฟังก์ชันการตรวจสอบการให้สิทธิในการทำงานของผู้ใช้ ด้วยการตรวจสอบรหัสของผู้ใช้และหมายเลขเครื่องที่ขอเข้าไปใช้ ขั้นตอนแรกในการใช้งานให้กดที่ตัวอักษร "**เริ่มการใช้งาน**" จะปรากฏหน้าจอ CheckPassword แสดงดังรูปที่ 5 ขึ้นมาให้กรอกข้อมูลต่างๆ ถ้าข้อมูลที่กรอกเข้าไปถูกต้องโปรแกรมจะทำการเข้าสู่การทำงานของหน้าต่างการแสดงผลในรูปแบบทั่วไปอย่างอัตโนมัติแสดงดังรูปที่ 6 และ 7



รูปที่ 5 หน้าต่างฟังก์ชันการตรวจสอบรหัสผู้ใช้

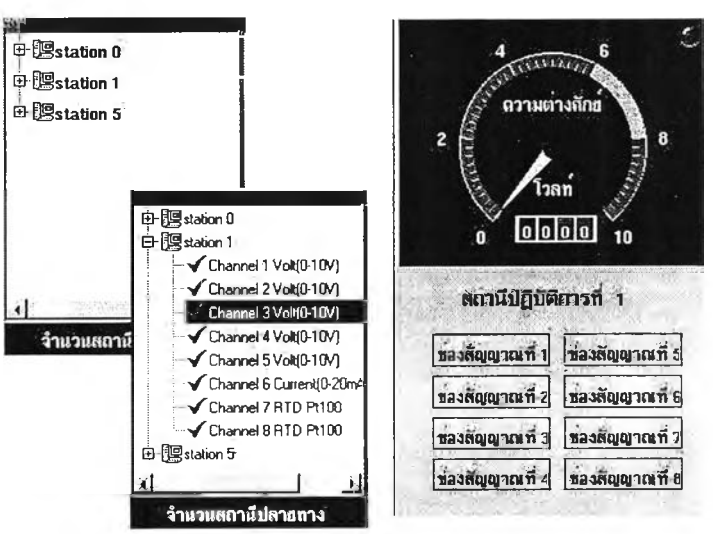
รูปที่ 6 หน้าต่างฟังก์ชันการตรวจสอบการติดต่อกับฐานข้อมูลของโปรแกรม Monitor



รูปที่ 7 หน้าต่างการแสดงผลในรูปแบบทั่วไปของโปรแกรม Monitor

ในส่วนการแสดงผลในรูปทั่วไปจะทำหน้าที่รายงานผลดังต่อไปนี้

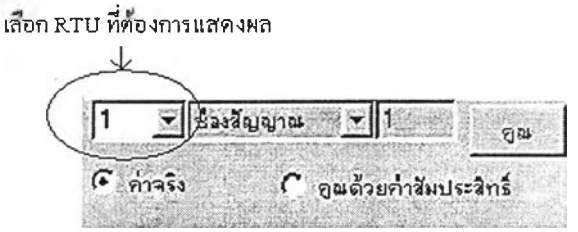
จำนวนสถานีปลายทาง เป็นส่วนแสดงผลการอ่านค่าจากฐานข้อมูล Onlnes ประเภทชนิดของอุปกรณ์ที่ติดตั้งกับสถานีหลักนำมาแสดงในรูปของจำนวน RTU ที่ติดตั้งกับสถานีหลักและรายละเอียดของช่องสัญญาณซึ่งเป็นส่วนของการแสดงผลอย่างเดียวไม่สามารถกำหนดประเภทและชนิดของช่องสัญญาณได้แสดงดังรูปที่ 8



รูปที่ 8 การรายงานจำนวน RTU และรายละเอียดของช่องสัญญาณ

หน้าจอแสดงผลในรูปแบบทั่วไป เป็นส่วนแสดงผลจากการอ่านข้อมูลของสถานีปลายทางระยะไกลนำมาแสดงในรูปของกราฟฟิกและตัวอักษรมีลักษณะการใช้งานเป็นลำดับขั้นดังนี้

- เลือก RTU ที่ต้องการแสดงผล และกำหนดรูปแบบการแสดงผลว่าให้แสดงค่าจริงที่รับเข้ามาหรือแสดงค่าของข้อมูลที่รับเข้ามาคูณด้วยค่าสัมประสิทธิ์



รูปที่ 9 การเลือก RTU ที่ต้องการแสดงผลและกำหนดรูปแบบการแสดงผล

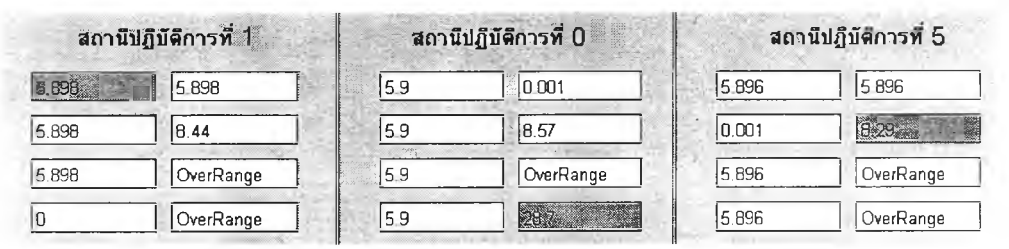
- เลือกช่องสัญญาณที่จะใช้แสดงผลโดยการตรวจสอบจาก "**จำนวนสถานีปลายทาง**" อย่างเช่นในรูปที่ 8 รูปแบบของการแสดงผลในรูปทั่วไปมีการแสดงผล 3 รูปแบบคือ
  - การแสดงผลในรูปแบบของสถานะของอินพุต / เอาต์พุต ดิจิตอล โดยที่แต่ละสถานีสามารถแสดงได้อย่างละ 4 ค่า ดังแสดงในรูปที่ 10

รูปที่ 10 หน้าต่างฟังก์ชันสำหรับแสดงสถานะของอินพุต / เอาต์พุต ดิจิตอล

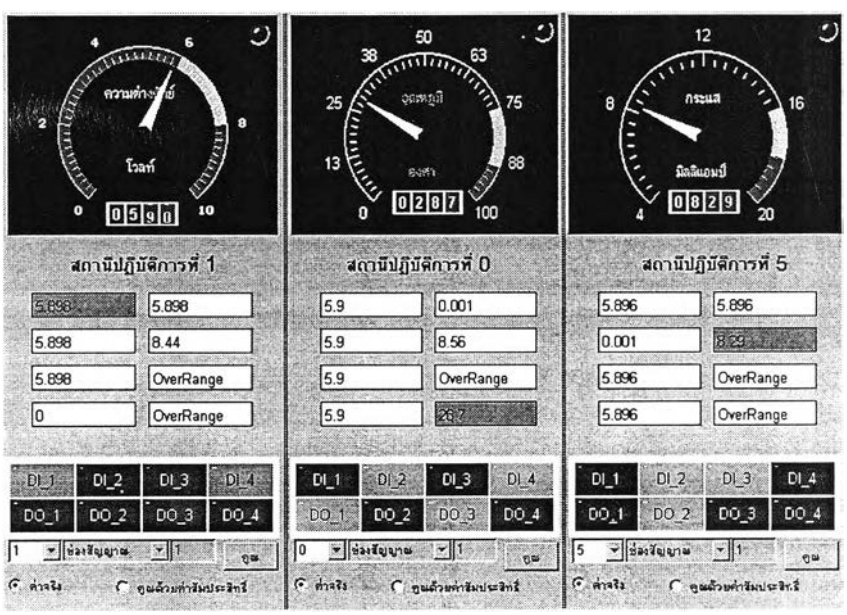
  - การแสดงผลในรูปแบบของมาตรวัดของข้อมูลแบบแอนะล็อก โดยจะแสดงผลได้ 3 แบบคืออุณหภูมิ มีหน่วยเป็นองศาเซลเซียส แรงดันมีหน่วยเป็นโวลต์ และ กระแสมีหน่วยเป็นมิลลิแอมป์ แสดงดังรูปที่ 11

รูปที่ 11 หน้าต่างฟังก์ชันสำหรับการแสดงผลในรูปแบบของมาตรวัดของข้อมูลแบบแอนะล็อก (Analog)

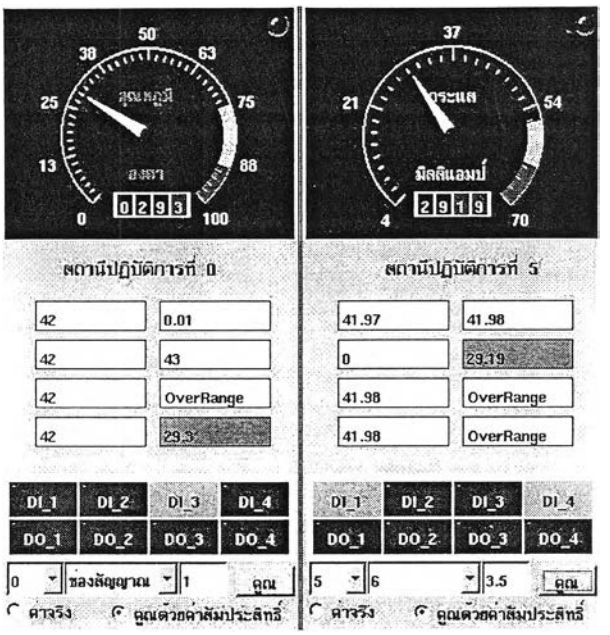
- การแสดงผลในรูปแบบของตัวอักษรของข้อมูลแบบแอนะล็อก



รูปที่ 12 หน้าต่างฟังก์ชันสำหรับการแสดงผลในรูปแบบของตัวอักษรของข้อมูลแบบแอนะล็อก (Analog)

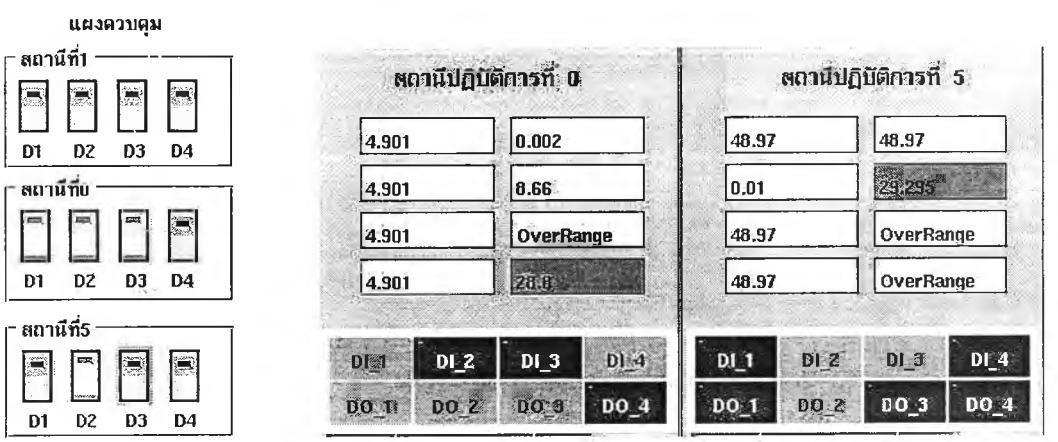


รูปที่ 13 ตัวอย่างการทำงานของหน้าต่างฟังก์ชันสำหรับการแสดงผลในรูปแบบทั่วไป เมื่อกำหนดให้ใช้ค่าจริงในการแสดงผล



รูปที่ 14 ตัวอย่างการทำงานของหน้าต่างฟังก์ชันสำหรับการแสดงผลในรูปแบบทั่วไป เมื่อกำหนดให้มีการใช้ค่าสัมประสิทธิ์คูณข้อมูลจริงก่อนการแสดงผล

แผงควบคุม เป็นฟังก์ชันการทำงานที่มีหน้าที่ควบคุมสถานีปลายทางระยะไกลโดยที่จะสามารถควบคุมสถานีใดได้บ้างนั้นขึ้นอยู่กับส่วนการกำหนดสถานีปลายทางระยะไกลที่ต้องการแสดงผลและสามารถเปลี่ยนแปลงสถานีที่ควบคุมได้ตลอดเวลา

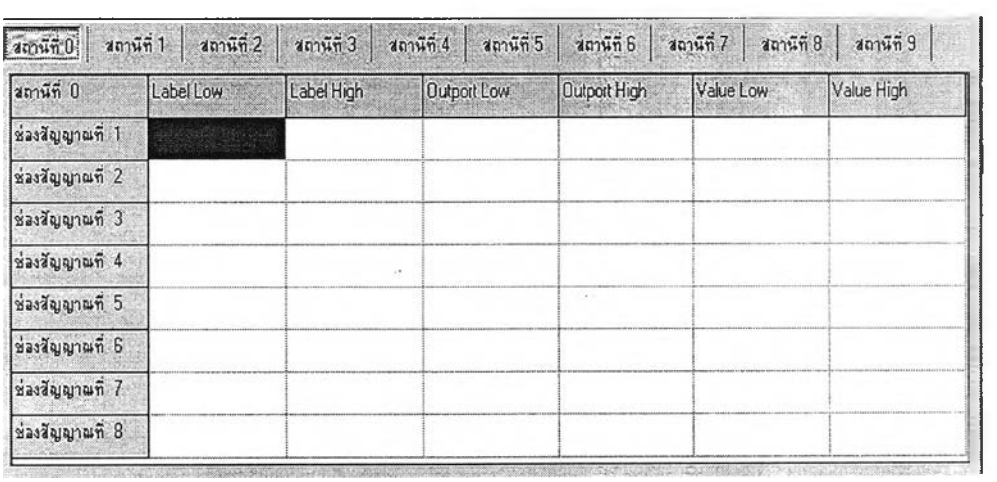


รูปที่ 15 หน้าต่างฟังก์ชันสำหรับการควบคุม RTU และการอ่านค่ากลับมาแสดงผล

กำหนดการเตือนข้อผิดพลาด เป็นส่วนที่ใช้กำหนดการแจ้งเตือนความผิดพลาด โดยสามารถแจ้งเตือนความผิดพลาดได้ 2 ระดับคือ High – Low ในช่องสัญญาณเดียวกัน นอกจากนั้นยังมีข้อกำหนดการแจ้งเตือนได้ 2 รูปแบบคือ การเตือนผ่านหน้าจอคอมพิวเตอร์ และการเตือนผ่านทางเอาต์พุตของสถานีปลายทางระยะไกล การกำหนดค่าการเตือนข้อผิดพลาดสามารถกำหนดได้ดังรูปที่ 16 ขั้นตอนแรกจะต้องทำการเลือกสถานีปฏิบัติการหรือ RTU ก่อน ต่อจากนั้นให้เลือกช่องสัญญาณที่ต้องการกำหนดการเตือนและใส่ค่าขอบเขตการเตือนโดยสามารถเลือกได้ว่าเป็นค่าขอบเขตที่มากกว่าหรือน้อยกว่าและถ้าต้องการให้แสดงผลออกที่พอร์ตดิจิตอลของ RTU นั้นด้วยให้กำหนดการเลือกช่องสัญญาณที่ Outport เป็นลำดับสุดท้ายแล้วกดปุ่มตกลงเพื่อยืนยันการกำหนดการแจ้งเตือน

รูปที่ 16 หน้าต่างฟังก์ชันสำหรับการกำหนดการแจ้งความผิดพลาด RTU ของโปรแกรม Monitor

หลังจากที่กำหนดการแจ้งเตือนข้อผิดพลาดเรียบร้อยแล้วสามารถตรวจสอบค่าที่กำหนดไว้ได้โดยกดที่ **"รายงานการกำหนดการแจ้งเตือนความผิดพลาด"**

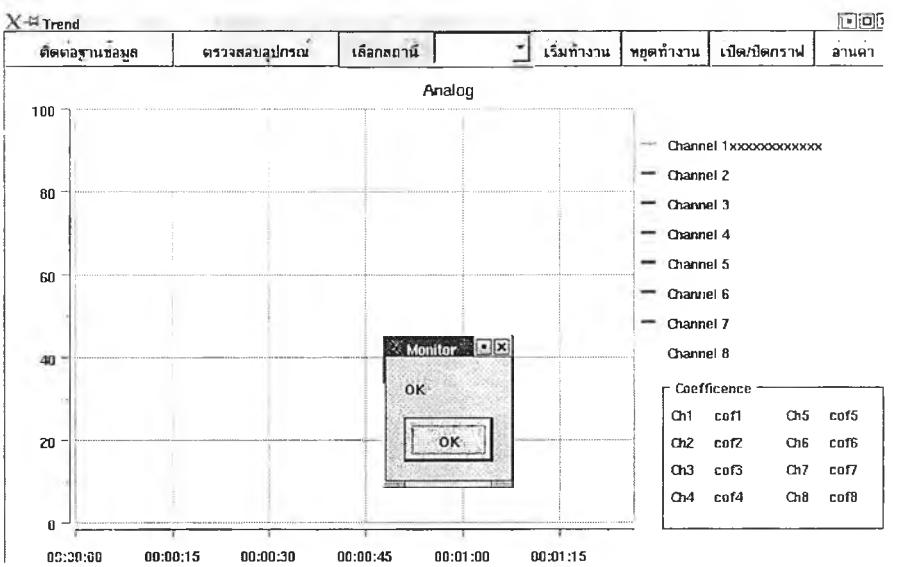


รูปที่ 17หน้าต่างรายงานการกำหนดการแจ้งเตือนความผิดปกติ

การแสดงการทำงานในรูปแบบกราฟ เป็นการเรียกใช้ฟังก์ชันการทำงานในรูปแบบของกราฟซึ่งเป็นการรายงานผลเทียบกับเวลาจริงซึ่งเรียกการทำงานได้จากการกดที่"**แสดงการทำงาน**" หลังจากนั้นจะปรากฏหน้าต่างดังรูปที่ 18 มีลักษณะการทำงานเป็นลำดับขั้นดังต่อไปนี้

ติดต่อฐานข้อมูล เป็นฟังก์ชันการทำงานตรวจสอบความพร้อมในการทำงานกับฐานข้อมูลของหน่วยแสดงผลแบบกราฟถ้าการตรวจสอบผลออกมาว่าไม่พร้อมก็จะไม่สามารถทำงานได้ แสดงดังรูปที่ 18

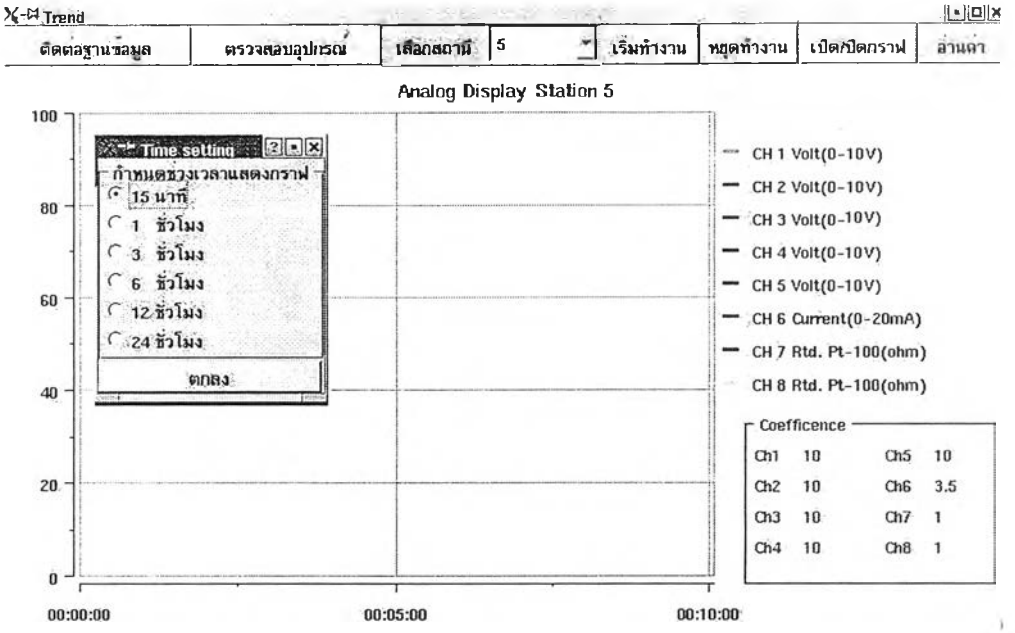
ตรวจสอบอุปกรณ์ เป็นฟังก์ชันการทำงานตรวจสอบชนิดของสัญญาณที่นำมาแสดงผลมีไว้เพื่อป้องกันความผิดพลาดในการอ่านข้อมูลของผู้ใช้



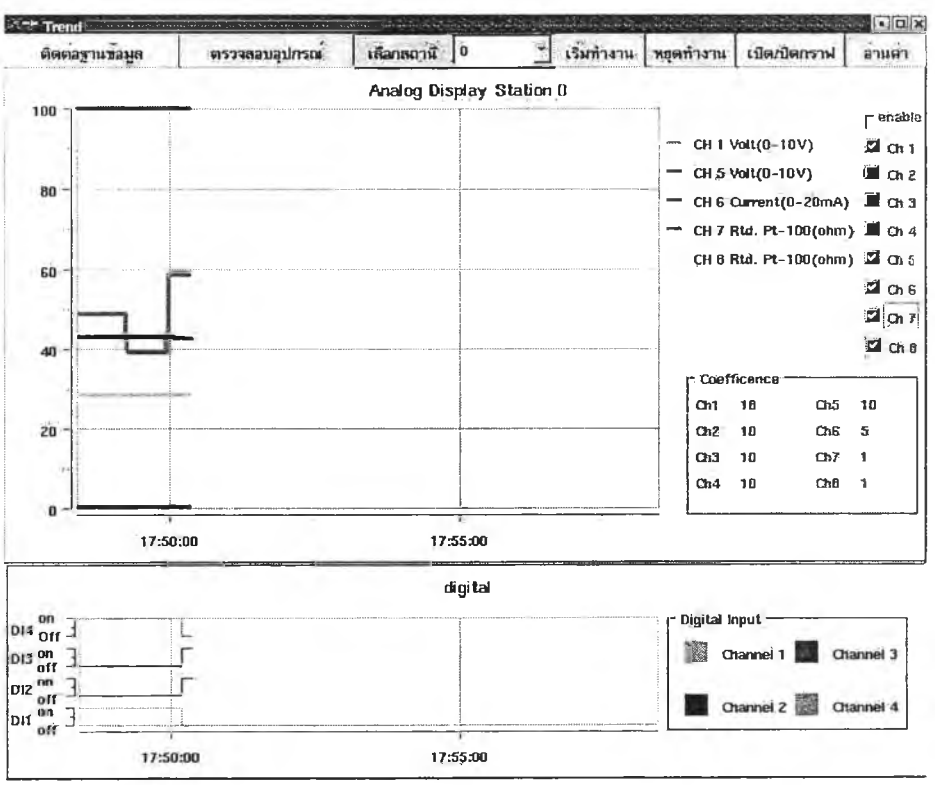
รูปที่ 18 หน้าต่างการตรวจสอบความพร้อมของฐานข้อมูลสำหรับการแสดงผล

เลือกสถานี เป็นฟังก์ชันการสำหรับเลือก RTU เข้ามาแสดงผลในรูปแบบของกราฟ แสดงได้ดังรูปที่ 20

เริ่มการทำงาน / หยุดการทำงาน เป็นฟังก์ชันสำหรับการเริ่มแสดงผลและหยุดการแสดงผล โดยขณะที่เริ่มการทำงานนั้นต้องมีการกำหนดระยะเวลาความกว้างของกราฟที่ใช้แสดงผลก่อน ดังรูปที่ 19



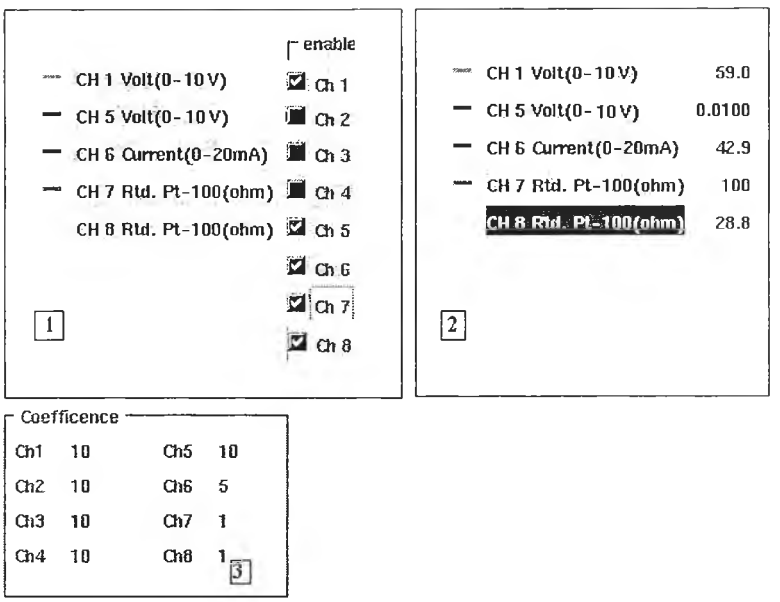
รูปที่ 19 หน้าต่างแสดงการกำหนดระยะเวลาความกว้างของกราฟในการอ่านข้อมูลจาก RTU



รูปที่ 20 หน้าต่างแสดงการทำงานในรูปแบบของกราฟ

เปิด/ปิดกราฟ เป็นฟังก์ชันที่ใช้ในการเลือกเปิดปิดช่องสัญญาณในการแสดงผล (1)

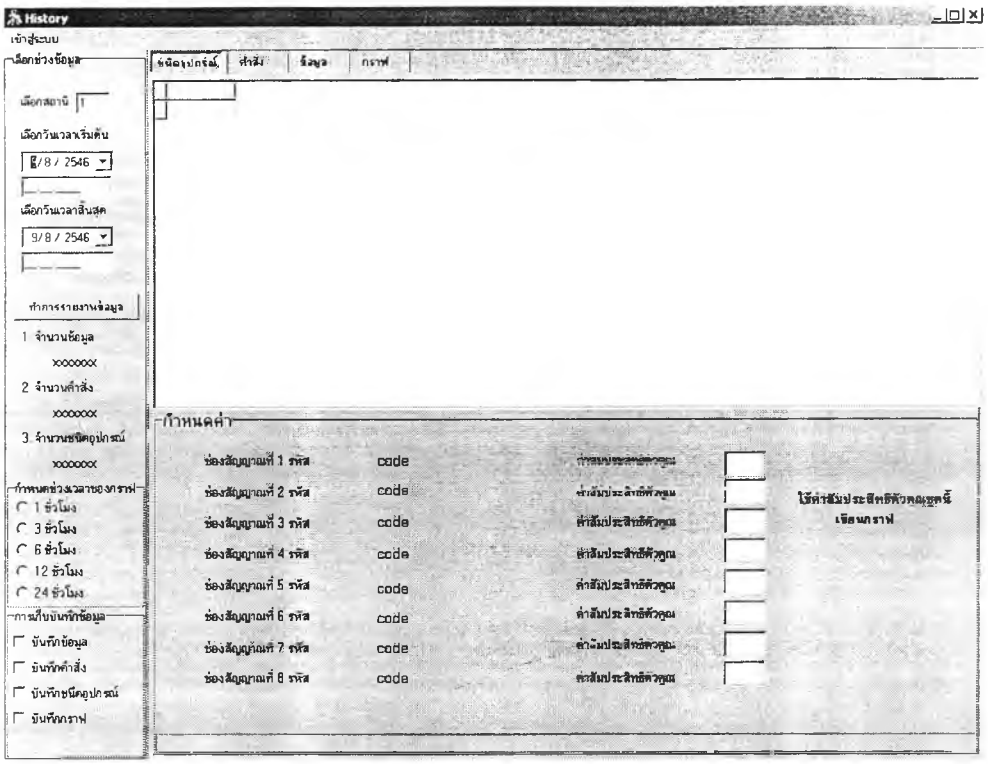
อ่านค่า เป็นฟังก์ชันที่ใช้ในการรายงานผลเป็นตัวเลขของแต่ละช่องสัญญาณ (2)



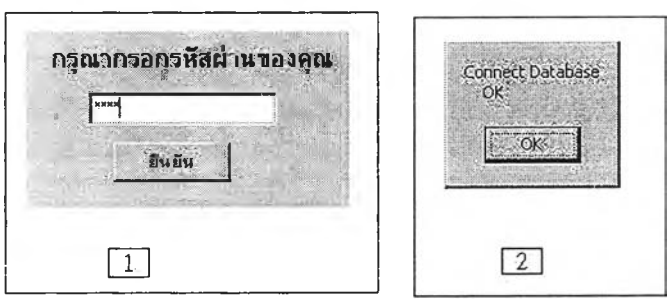
รูปที่ 21 การใช้งานฟังก์ชัน เปิด/ปิด กราฟ และการอ่านค่า

แสดงฐานประวัติข้อมูล เป็นฟังก์ชันการดึงข้อมูลจากฐานข้อมูลประวัตินำมาแสดงในรูปแบบของประวัติของชนิดอุปกรณ์ ประวัติของของชุดคำสั่งและประวัติของข้อมูลแอนะล็อกและดิจิตอลเทียบกับเวลาในขณะที่ทำการบันทึกข้อมูลไว้รวมถึงการเขียนกราฟและดึงข้อมูลออกมาในรูปของไฟล์เอกสารเพื่อนำมาใช้การทำรายงาน แสดงได้ดังรูปที่ 22 โดยมีรายละเอียดของส่วนแสดงฐานประวัติข้อมูลมีดังต่อไปนี้

ติดต่อฐานข้อมูล เป็นฟังก์ชันการทำงานตรวจสอบความพร้อมในการทำงานกับฐานข้อมูลของหน่วยแสดงผลแบบฐานประวัติข้อมูลถ้าการตรวจสอบผลออกมาว่าไม่พร้อมก็จะไม่สามารถทำงานได้ แสดงดังรูปที่ 23

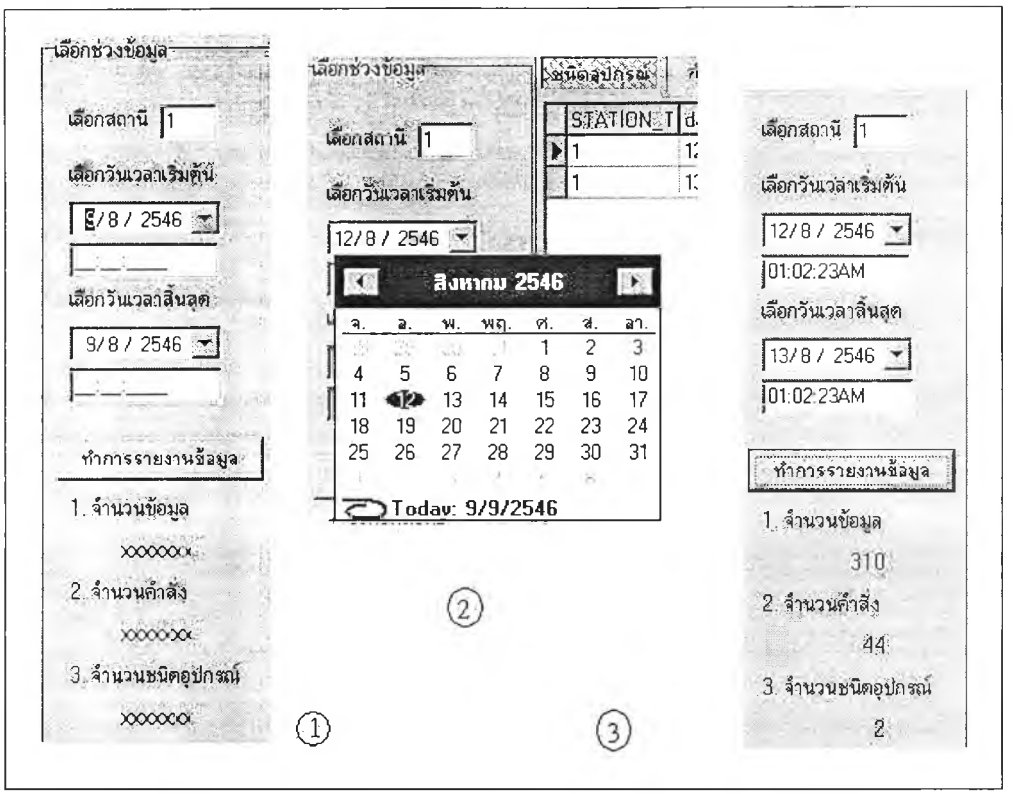


รูปที่ 22 หน้าต่างของส่วนแสดงฐานประวัติข้อมูล



รูปที่ 23 หน้าต่างการตรวจสอบความพร้อมของฐานข้อมูลสำหรับการแสดงผล

เลือกช่วงข้อมูล เป็นฟังก์ชันที่ใช้ในการเลือกสถานีและช่วงเวลาที่ทำการแสดงผล ประกอบไปด้วยการทำงานต่างๆที่แสดงดังรูปที่ 24



รูปที่ 24 การอ่านข้อมูลประเภทชนิดอุปกรณ์จากฐานข้อมูลประวัติ

จากการกำหนดรายละเอียดการอ่านข้อมูลจากฐานข้อมูลประวัติจะได้ข้อมูลออกมาดังแสดงในรูปที่ 25

ชนิดอุปกรณ์ | คำสั่ง | ข้อมูล | กราฟ

ID_Command	STATION_C	STATUS_C	datetimes_C	Active_DO1	Active_DO2	Active_DO3	Active_DO4
2	1	A	12/8/2003 22:25:09	0	0	1	0
2	1	A	12/8/2003 22:25:14	0	0	0	0
2	1	A	12/8/2003 22:26:03	0	1	0	0
2	1	A	12/8/2003 22:26:09	0	0	0	0
2	1	A	13/8/2003 0:30:41	0	0	1	0
2	1	A	13/8/2003 0:30:42	0	1	1	0
2	1	A	13/8/2003 0:31:05	0	0	1	0
2	1	A	13/8/2003 0:31:12	0	1	1	0
1	1	A	13/8/2003 0:32:39	1	0	0	0

รูปที่ 25 การอ่านข้อมูลประเภทคำสั่งจากฐานข้อมูลประวัติ

ชนิดอุปกรณ์ | คำสั่ง | ข้อมูล | กราฟ

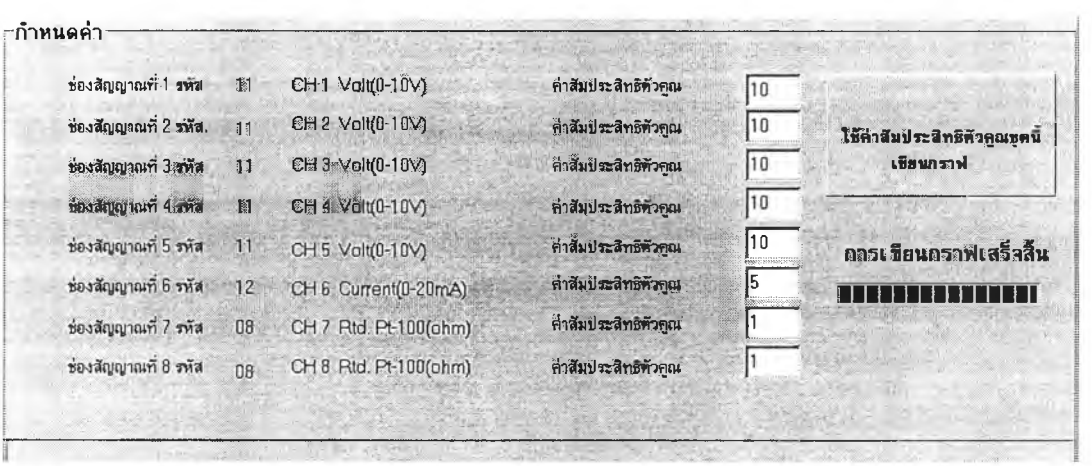
STATION_T	datetimes_T	AT1	AT2	AT3	AT4	AT5	AT6	AT7	AT8	COF1	COF2	COF3	COF4	COF5	COF6	COF7	COF8
1	12/8/2003 22:37:15	11	11	11	11	11	12	08	08	10	10	10	10	10	5	1	1
1	13/8/2003 2:02:18	11	11	11	11	11	12	08	08	10	10	10	10	10	5	1	1

ชนิดอุปกรณ์ | คำสั่ง | ข้อมูล | กราฟ

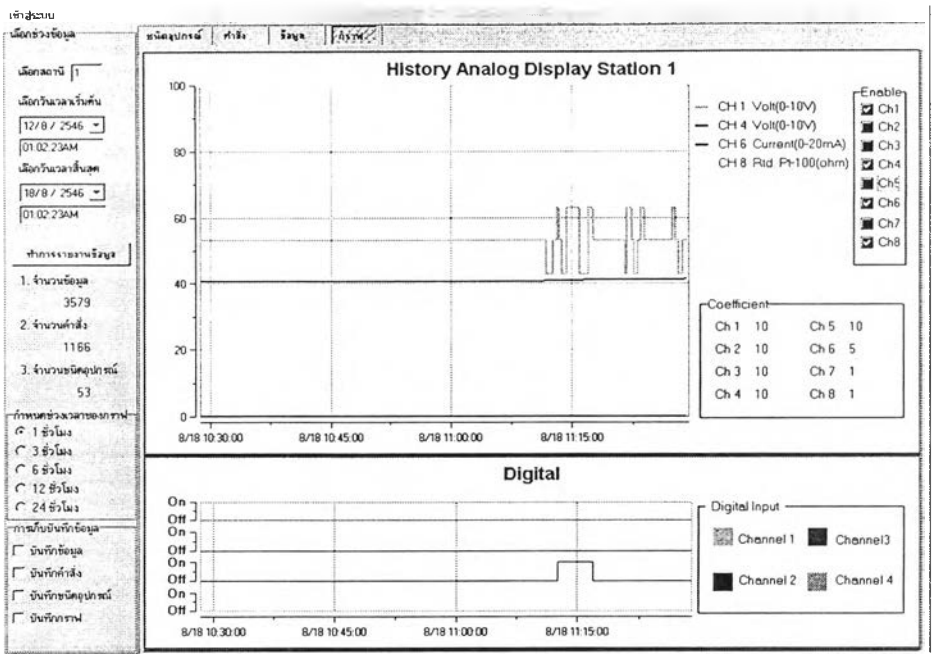
Dates	Station_Num	Analog_In1	Analog_In2	Analog_In3	Analog_In4	Analog_In5	Analog_In6	Analog_In7	Analog_In8	Digital_In1	Digital_In2
12/8/2003 22:24:17	1	6.000	5.999	5.999	0.000	5.999	8.13	0	0	0	0
12/8/2003 22:24:23	1	5.999	5.999	5.999	0.000	5.999	8.13	0	0	0	0
12/8/2003 22:24:28	1	5.999	5.999	5.999	-0.000	5.999	8.13	0	0	0	0
12/8/2003 22:24:34	1	5.999	5.999	5.999	0.001	5.999	8.13	0	0	0	0
12/8/2003 22:24:39	1	5.999	5.999	5.999	0.000	5.999	8.13	0	0	0	0
12/8/2003 22:24:45	1	6.000	6.000	6.000	0.000	5.999	8.13	0	0	0	0
12/8/2003 22:24:50	1	5.999	5.999	5.999	0.000	5.999	8.13	0	0	0	0
12/8/2003 22:24:56	1	5.999	5.999	5.999	0.000	5.999	8.13	0	0	0	0
12/8/2003 22:25:01	1	5.999	5.999	5.999	0.000	5.999	8.14	0	0	0	0
12/8/2003 22:25:07	1	5.999	5.999	5.999	0.000	5.999	8.14	0	0	0	0

รูปที่ 25 (ต่อ) การอ่านข้อมูลประเภทคำสั่งจากฐานข้อมูลประวัติ

กำหนดค่า เป็นฟังก์ชันที่ใช้ในการกำหนดค่าการเขียนกราฟโดยที่ผู้ใช้งานสามารถเปลี่ยนแปลงแก้ไขสัมประสิทธิ์ตัวคูณในการเขียนกราฟได้แต่ถ้าไม่มีการกำหนดการเปลี่ยนแปลงค่าสัมประสิทธิ์ตัวคูณ โปรแกรมจะค้นหาชนิดมูลและเขียนกราฟเทียบเป็น 100 เปอร์เซนต์ทั้งหมด นอกจากนี้แล้วฟังก์ชันการทำงานนี้ยังใช้ตรวจสอบความต่อเนื่องของข้อกำหนดของช่องสัญญาณที่วัดคือถ้าในกรณีที่สถานีเดียวกันมีการเปลี่ยนแปลงชนิดของช่องสัญญาณ เช่น เปลี่ยนจากการวัดกระแส เป็นวัดอุณหภูมิแทนถ้านำค่าสัมประสิทธิ์ตัวคูณชนิดเดียวกันมาเขียนกราฟทั้งหมดจะทำให้การอ่าข้อมูลผิดพลาดได้ การกำหนดค่าสัมประสิทธิ์ตัวคูณและการเขียนกราฟแสดงได้ดังรูปที่ 26 และ 27

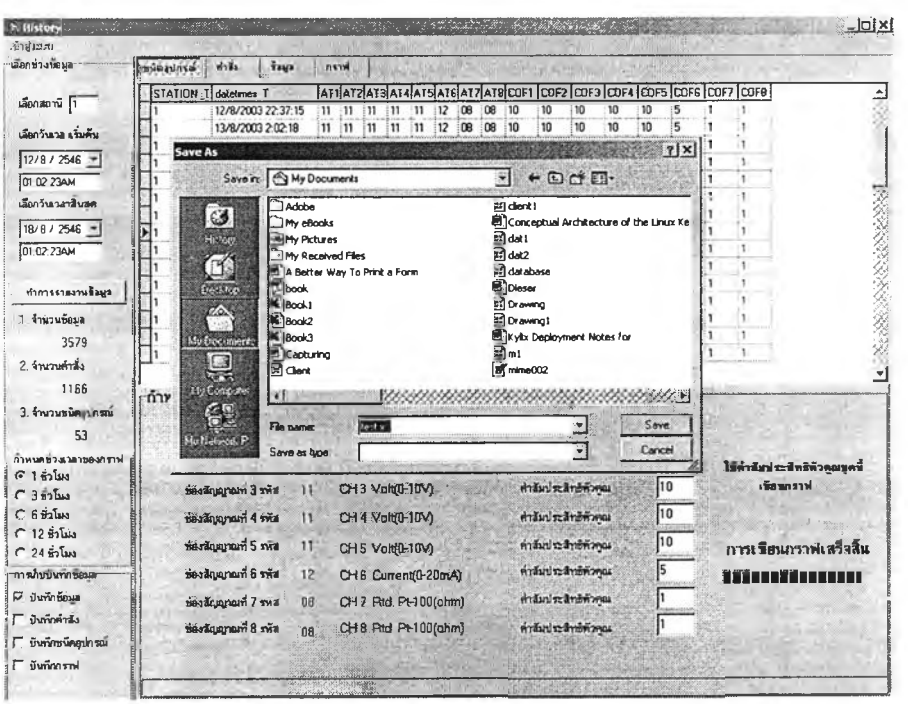


รูปที่ 26 การกำหนดค่าสัมประสิทธิ์ตัวคูณที่ใช้ในการเขียนกราฟ



รูปที่ 27 การอ่านข้อมูลประเภทข้อมูลแอนะล็อกและดิจิตอลจากฐานข้อมูลประวัติในรูปแบบของกราฟ

การเก็บบันทึกข้อมูล เป็นฟังก์ชันการดึงข้อมูลจากฐานข้อมูลมาเก็บได้ 2 รูปแบบคือ ในรูปแบบของไฟล์เอกสาร และ รูปกราฟ มีรายละเอียดการทำงานดังรูปที่ 28



รูปที่ 28 การบันทึกข้อมูลในรูปของไฟล์เอกสาร

ภาคผนวก ง
บทความที่ได้รับการตีพิมพ์จากการประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 26

ระบบสกาดาบนระบบปฏิบัติการลีนุกซ์

A LINUX BASED SCADA SYSTEM

ชนวรรณ เกตุดี กฤษดา วิศวธีรานนท์ มานพ วงศ์สายสุวรรณ
ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ถนนพญาไท เขตปทุมวัน กรุงเทพมหานคร 10330
โทร 0-2218-6515 โทรสาร 0-2218-8991 E-mail :44706945@student.chula.ac.th

บทคัดย่อ

บทความนี้เป็นการนำเสนอการพัฒนาระบบสกาดาซึ่งเป็นระบบวัดและบันทึกข้อมูลในอุตสาหกรรมบนระบบปฏิบัติการลีนุกซ์ โดยระบบมีคุณสมบัติในการรวบรวมข้อมูลจาก RTU (Remote Terminal Unit) นำมาแสดงผลการทำงานในขณะเวลาจริงพร้อมทั้งการเตือนความผิดปกติในรูปแบบที่สังเกตุได้ง่าย รวมถึงการบันทึกข้อมูลลงในฐานข้อมูลเพื่อนำไปใช้ในรูปของเอกสารหรือให้บริการทางด้านเครือข่ายในรูปแบบของระบบรับ-ให้บริการ (Client-Server) ผลจากการพัฒนาพบว่าการพัฒนาระบบสกาดาบนระบบปฏิบัติการลีนุกซ์ไม่ได้มีความยุ่งยากและมีเครื่องมือสนับสนุนในการพัฒนามากมาย ระบบที่ได้มีเสถียรภาพ ราคาถูกและมีคุณภาพเทียบเท่าได้กับสกาดาบนระบบปฏิบัติการไมโครซอฟต์วินโดว์

Abstract

This paper presents the development of SCADA system on Linux Operating System (OS). The Linux SCADA has equivalent performances which can replace on MS-Windows. It is able to collect data from RTU (Remote Terminal Unit) and display in real-time. Furthermore, It contains various functions of SCADA system. Users can use the actual monitoring applications on MS-Windows or Linux OS through client-server network service. Moreover, The Linux SCADA can be easily developed since there are many available supporting tools from the Free Software Foundation . In addition, the system stability can be provided with reasonable prices.

Kewords : Linux, SCADA

บทนำ

ระบบสกาดา (SCADA) เป็นระบบวัดและบันทึกข้อมูลที่มีใช้กันมากในอุตสาหกรรม ใช้ในการควบคุมกระบวนการผลิตและเชื่อมโยงข้อมูลการผลิตเพื่อรายงานข้อมูลไปยังส่วนบริหารโรงงาน ในระบบสกาดาขนาดใหญ่ที่มีเสถียรภาพสูงส่วนใหญ่จะทำงานบนระบบปฏิบัติการยูนิกซ์ ส่วนในระบบสกาดาที่ใช้กันทั่วไปส่วนใหญ่ทำงานอยู่บนระบบปฏิบัติการไมโครซอฟต์วินโดว์ซึ่งมีปัญหาบ่อยครั้งเมื่อไมโครซอฟต์วินโดว์เปลี่ยนรุ่น (version) การเปลี่ยนรุ่นจะส่งผลต่อการทำงานของระบบสกาดาที่มีอยู่เดิมในบางส่วนหรือส่งผลให้ไม่สามารถใช้งานได้จึงจำเป็นต้องพัฒนาใหม่หรือเสียค่าใช้จ่ายเพื่อปรับปรุงให้กลับมาใช้ได้ดังเดิมและอาจจะต้องเปลี่ยนเครื่องมือที่จะต้องใช้พัฒนาใหม่ด้วย ซึ่งเป็นอุปสรรคที่สำคัญของการขาดความต่อเนื่องในการพัฒนาระบบสกาดาของเดิมที่มีอยู่ให้มีประสิทธิภาพและทันสมัยอยู่ตลอดเวลา

ดังนั้นบทความนี้จึงนำเสนอการพัฒนาระบบสกาดาบนระบบปฏิบัติการลีนุกซ์ซึ่งมีการเปิดเผยรหัสต้นฉบับ (Open Source) ดังนั้นเมื่อต้องการปรับปรุงแก้ไขหรือเพิ่มเติมก็จะสามารถทำได้สะดวกเพราะมีแหล่งข้อมูลและเอกสารที่เปิดเผยและหาได้ง่ายสนับสนุนมากมายโดยไม่ต้องเสียค่าลิขสิทธ์ทางซอฟต์แวร์ และเนื่องจากระบบปฏิบัติการลีนุกซ์ถูกถอดแบบมาจากระบบปฏิบัติการยูนิกซ์ซึ่งมีชื่อเสียงทางประสิทธิภาพและเสถียรภาพสูง [3] จึงทำให้ระบบสกาดาที่ถูกพัฒนาด้วยระบบปฏิบัติการลีนุกซ์มีความเชื่อถือได้ การพัฒนาระบบสกาดาในบทความนี้มุ่งเน้นการออกแบบระบบให้มีความครบถ้วนการใช้งานต่างๆของระบบสกาดาขนาดกลางและมีความสามารถพิเศษในการที่จะติดต่อกับ RTU (Remote Terminal Unit) หรือตัวเก็บข้อมูลได้หลายชนิดและหลายตัวตามมาตรฐาน RS-485 โดยอาศัยการเพิ่มเติมบางส่วนของระบบเท่านั้น ทั้งนี้ก็เพื่อให้สามารถนำไปใช้งานได้จริงในงานอุตสาหกรรม

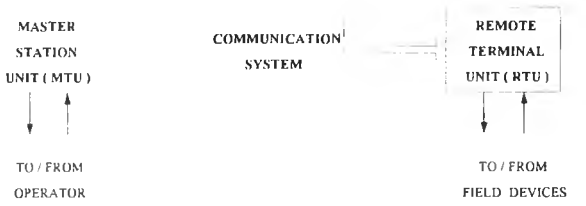
1. สกาดา (SCADA)

ย่อมาจากคำว่า Supervisory Control And Data Acquisition หมายถึง "ระบบ" ที่มีการจัดการ

- รวบรวมข้อมูลจากที่ต่าง ๆ (Collection of Information)
- ส่งไปที่ศูนย์ควบคุม (Transferring Data to a Central Site)
- แสดงสถานะการทำงานของอุปกรณ์ในระบบ (Monitoring)
- บันทึกไว้เป็นฐานข้อมูล (Data Acquisition)
- วิเคราะห์และประมวลผล (Analysis and Data Processing) ซึ่งหมายถึงการใช้ คอมพิวเตอร์ เข้ามาช่วยในการวิเคราะห์
- ส่งผลไปควบคุมการทำงานของอุปกรณ์ในระบบได้ (Control)

2. โครงสร้างระบบสกาดา

ในกรณีทั่วไปอุปกรณ์ควบคุมและจัดหาข้อมูลจะประกอบด้วย สถานีหลัก (Master Station Unit) อย่างน้อย 1 สถานีหน่วยควบคุมปลายทางระยะไกล (RTU) ของสถานีปฏิบัติการ (Field Sites) หนึ่งตัวหรือมากกว่า และมีการเก็บข้อมูลหรือการควบคุมระหว่างสถานีหลักกับหน่วยควบคุมปลายทางระยะไกลโดยการส่งผ่านระบบการสื่อสาร ดังแสดงในรูปที่ 1



รูปที่ 1 โครงสร้างพื้นฐานของระบบสกาดา

ตามหลักการของระบบ สกาดา ([1],[2] อ้างอิงตาม IEEE Standard Definition 37.1 – 1994) จะต้องมีคุณสมบัติโดยสังเขปดังต่อไปนี้

2.1 สถานีหลัก (Master Terminal Unit)

คือจุดรวมของอุปกรณ์ทุกอย่างครบถ้วน ฟังก์ชันการทำงานและอุปกรณ์อื่นๆที่เชื่อมต่อกันทางไฟฟ้าเพื่อทำงานตามลักษณะฟังก์ชันควบคุมของสถานีควบคุมหลัก อุปกรณ์ทั้งหมดนี้รวมไปถึงจุดเชื่อมต่อกับช่องสัญญาณสื่อสาร แต่ไม่รวมช่องสัญญาณที่เชื่อมต่อกัน ระหว่างการติดต่อสื่อสารกับสถานีควบคุมทางไกลอื่นๆ สถานีหลักจะเป็นจุดที่มีความสำคัญที่สุดในระบบสื่อสาร สามารถแบ่งตามลักษณะการทำงานเป็น 2 แบบ คือ

2.1.1 สถานีหลักแบบรวมฟังก์ชันไว้ที่ศูนย์กลาง คือการรวมฟังก์ชันทำงานต่างๆที่ใช้ในระบบสกาดามารวมไว้ที่เครื่องคอมพิวเตอร์เครื่องเดียว (อาจมีการสำรองข้อมูลไว้อีกหนึ่งเครื่องก็ได้)

2.1.2 สถานีหลักแบบกระจายฟังก์ชัน คือมีการแบ่งฟังก์ชันการทำงานเป็นกลุ่มๆกระจายออกไปตามหน้าที่ที่ทำงานโดยแบ่งไปอยู่บน PC คนละเครื่อง เช่น เครื่องหนึ่งสำหรับแสดงผลอีกเครื่องหนึ่งสำหรับควบคุม และอีกเครื่องหนึ่งสำหรับเก็บข้อมูลเพื่อการวิเคราะห์ เป็นต้น

2.2 หน่วยควบคุมปลายทางระยะไกล (RTU)

คือจุดรวมของอุปกรณ์ทุกอย่างครบถ้วน ฟังก์ชันการทำงานและอุปกรณ์อื่นๆที่เชื่อมต่อกันทางไฟฟ้าเพื่อทำงานตามลักษณะฟังก์ชันควบคุมของสถานีควบคุมทางไกล อุปกรณ์ทั้งหมดนี้รวมไปถึงจุดเชื่อมต่อกับช่องสัญญาณสื่อสาร แต่ไม่รวมช่องสัญญาณที่เชื่อมต่อกัน ระหว่างการติดต่อสื่อสารกับสถานีหลัก

2.3 ระบบสื่อสาร (Communication System)

มีหน้าที่หลักในการเชื่อมโยงระหว่างกลุ่มของสถานีหลักกับกลุ่มของหน่วยควบคุมปลายทางระยะไกล ซึ่งคุณสมบัติโดยสังเขปต้องมีดังต่อไปนี้เป็นอย่างน้อย

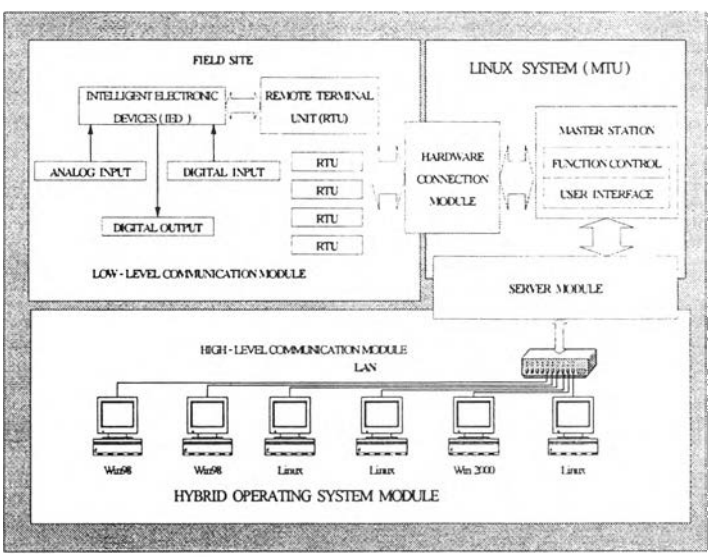
2.3.1 มีการกำหนดการใช้อุปกรณ์ในการรับส่งข้อมูล เช่น โมเด็ม (Modem), วิทยุ (ในย่าน UHF) เป็นต้น

2.3.2 มีการกำหนดช่องสัญญาณในการรับส่งข้อมูล

2.3.3 มีสื่อกลางการสื่อสาร(โพรโตคอล) เช่น MODBUS [7] , ASCII , RS-485 เป็นต้น

3. การออกแบบโครงสร้างลีนุกซ์สกาดา

การออกแบบระบบสกาดาบนระบบปฏิบัติการลีนุกซ์ที่นำมาทดสอบการทำงานถูกออกแบบให้เป็นระบบสกาดาแบบสถานีหลัก (Master Station Unit) เป็นแบบรวมฟังก์ชันไว้ที่ศูนย์กลางและมีการติดต่อกับสถานีปฏิบัติการ (Field Site) ซึ่งเป็นแหล่งรวมหน่วยควบคุมปลายทางระยะไกล (RTU) โดยสื่อสารกันด้วยมาตรฐาน RS-485 โดยการโพลลิ่ง (Polling) ด้วยโพรโตคอลสื่อสารแบบ MODBUS (ASCII) และสามารถตรวจสอบสถานะการทำงาน (Monitoring) ได้โดยผ่านระบบปฏิบัติการไมโครซอฟต์วินโดว์และลีนุกซ์ ดังแสดงไว้ตามรูปที่ 2



รูปที่ 2 โครงสร้างที่ใช้ทดสอบการทำงานของลีนุกซ์สกาดา

รายละเอียดโครงสร้างลีนุกซ์สกาดา

3.1 ระบบลีนุกซ์ (Linux System) เป็นระบบที่มีหน้าที่ควบคุมและบริหารหน่วยทำงานย่อยดังต่อไปนี้

3.1.1 สถานีหลัก (Master Station Unit) ประกอบด้วย

- ส่วนติดต่อประสานกับผู้ใช้ (User Interface) ทำหน้าที่ในการตอบสนองความต้องการกับผู้ใช้ตามคุณสมบัติของระบบสกาดา

- ฟังก์ชันควบคุม (Function Control) มีหน้าที่ในการรับคำสั่งจากส่วนติดต่อส่วนประสานงานกับผู้ใช้เพื่อที่จะใช้จัดการเกี่ยวกับระบบฐานข้อมูลและการรับข้อมูลจากหน่วยควบคุมปลายทางระยะไกล (RTU) เช่น การเก็บข้อมูลเข้าฐานข้อมูล, การแสดงผลในรูปกราฟฟิกหรือตัวอักษรตลอดจนการสั่งให้หน่วยควบคุมปลายทางระยะไกลทำงานตามคำสั่งของผู้ควบคุมระบบ

3.1.2 หน่วยเชื่อมโยงฮาร์ดแวร์ (Hardware Connection Module) มีหน้าที่ในการควบคุมการรับส่งข้อมูลและแปลงข้อมูลจากข้อมูลดิบเป็นข้อมูลที่สามารถทำไปใช้ในรูปแบบต่างๆได้

3.1.3 หน่วยบริการเครือข่าย (Server Module) ทำหน้าที่ในการนำข้อมูลจากฐานข้อมูลในสถานีหลัก (Master Station) ออกมาเผยแพร่ให้หน่วยระบบปฏิบัติการแบบผสมผสาน (Hybrid Operating System Module) นำไปใช้ได้

3.2 สถานีปฏิบัติการ (Field Site) ประกอบด้วย

หน่วยควบคุมปลายทางระยะไกล (RTU) ทำหน้าที่ในการติดต่อควบคุมกับการติดต่อสื่อสารอุปกรณ์ปลายทางฉลาด (Intelligent Electronic Devices) ซึ่งรวมถึงอุปกรณ์ที่รับส่งสัญญาณแอนะล็อก (Analog Signal) และสัญญาณดิจิทัล (Digital Signal) ด้วยเพื่อรับส่งข้อมูลให้กับสถานีหลัก ในการทดสอบนี้ได้เลือกอุปกรณ์ที่มีผู้พัฒนาแล้วและมีจำหน่ายอยู่มาเป็นหน่วยควบคุมปลายทางระยะไกล (RTU) โดยอุปกรณ์ที่เลือกมาใช้ต้องเป็นการประยุกต์ใช้งานโพรโตคอลสื่อสารแบบ MODBUS [4]

3. 3 หน่วยสื่อสาร (Communication Module) แบ่งเป็น 2 ระดับคือ

การติดสื่อสารระดับล่าง (Low-Level Communication Module) คือ การติดต่อสื่อสารระหว่างหน่วยควบคุมปลายทางระยะไกล (RTU) กับสถานีปฏิบัติการหลักโดยสื่อสารกันด้วยมาตรฐาน RS-485 โดยการโพลลิ่ง (Polling)

การติดสื่อสารระดับบน (High-level Communication Module) คือ การติดต่อสื่อสารระหว่างระบบลีนุกซ์ (Linux System) กับหน่วยระบบปฏิบัติการแบบผสมผสาน (Hybrid Operating System Module) โดยผ่านหน่วยบริการเครือข่าย (Server Module) โดยใช้โพรโตคอล TCP/IP เป็นมาตรฐานในการติดต่อสื่อสาร

3.4 หน่วยระบบปฏิบัติการแบบผสมผสาน (Hybrid Operating System Module) เป็นส่วนของโปรแกรมลูกข่ายที่สามารถทำงานของระบบสกาดาได้เกือบทุกอย่างยกเว้นเพียงการควบคุมหน่วยควบคุมปลายทางระยะไกลเท่านั้นและยังทำงานได้บนระบบปฏิบัติการไมโครซอฟต์วินโดว์และลีนุกซ์

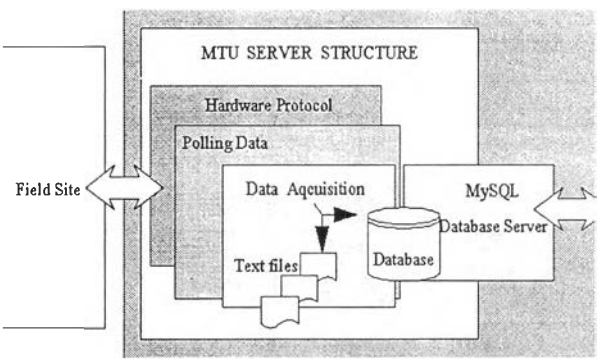
4. การพัฒนา

หลังจากกำหนดโครงสร้างระบบลีนุกซ์ชัดเจนแล้ว ผู้พัฒนาได้พัฒนาซอฟต์แวร์ขึ้นมาทดสอบดังโครงสร้างดังต่อไปนี้

4.1 โครงสร้างการเก็บและให้บริการข้อมูล เป็นการนำระบบปฏิบัติการลีนุกซ์มาประยุกต์ใช้งานแบบ Data Acquisition [5] มีการแบ่งการทำงาน ดังต่อไปนี้

Hardware Protocol

เป็นส่วนของโปรแกรมที่ใช้บรรจุโพรโตคอลการสื่อสารข้อมูลที่ถูกเรียกใช้โดยการโพลลิ่งข้อมูลโดยการทดสอบนี้เลือกใช้ Modbus ASCII บนมาตรฐานการส่งสัญญาณแบบ RS-485 และเราสามารถเพิ่มการเรียกใช้มาตรฐานการส่งสัญญาณแบบอื่นได้ในส่วนนี้ โดยลักษณะการสื่อสารแสดงได้ตาม รูปที่ 4 ในส่วนของการติดต่อระหว่าง Linux Server PC กับ Field Site



รูปที่ 3 โครงสร้างการเก็บและให้บริการข้อมูล [8]

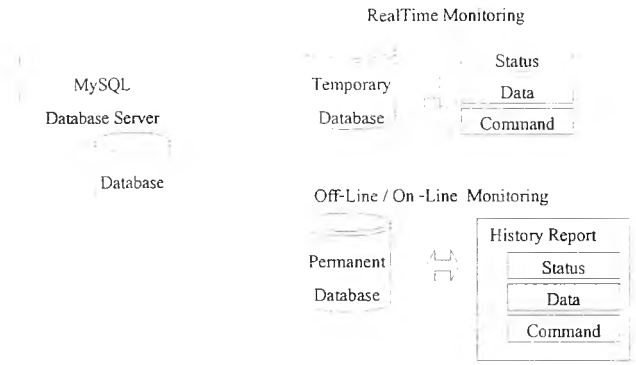
รูปที่ 4 รูปแบบของการสื่อสารข้อมูล

Polling Data

ในส่วนนี้จะหน้าที่ดึงข้อมูลจาก RTU ขึ้นมาตามโพรโตคอลที่กำหนดไว้รวมทั้งการตรวจสอบสถานะภาพการติดต่อระหว่าง RTU กับ LINUX SERVER ว่ามีสิ่งผิดปกติหรือเปล่า เช่นสายหลุดเป็นต้น รวมทั้งการตรวจสอบข้อมูลที่รับเข้ามาว่ามีความถูกต้องตามรูปแบบที่กำหนดก่อนที่จะนำข้อมูลบันทึกลงใน Database ต่อไป

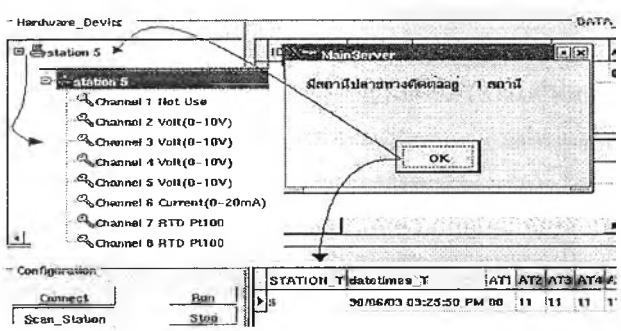
Data Acquisition

เป็นการส่วนของการนำข้อมูลที่ผ่านการตรวจสอบแล้วทำการบันทึกลงใน Database 2 ประเภท ดังแสดงในรูปที่ 5 คือ



รูปที่ 5 รูปแบบของการเก็บบันทึกข้อมูล

1. Temporary Database ข้อมูลที่อยู่ใน Database ชนิดนี้จะมีอายุการใช้งานเพียงชั่วคราวเท่านั้นเพื่อจะส่งข้อมูลต่อให้กับส่วนแสดงผลในรูปเทียบกับเวลาจริง (RealTime) เป็นหลักและแต่ละชนิดของข้อมูลก็มีอายุใช้การงานไม่เท่ากันเช่น ข้อมูลชนิด Status (สถานะการติดต่อกับ RTU) จะคงอยู่เฉพาะตอนเรียกตรวจสอบประเภทของข้อมูลที่ได้รับเข้ามา หลังจากเรียกตรวจสอบใหม่ก็จะเปลี่ยนแปลงไปตามค่าที่เรากำหนดไว้ ดังแสดงในรูปที่ 6 ส่วน Command (คำสั่ง) ก็จะถูกลบทันทีที่ Linux Server ปฏิบัติงานตามคำสั่งเสร็จ



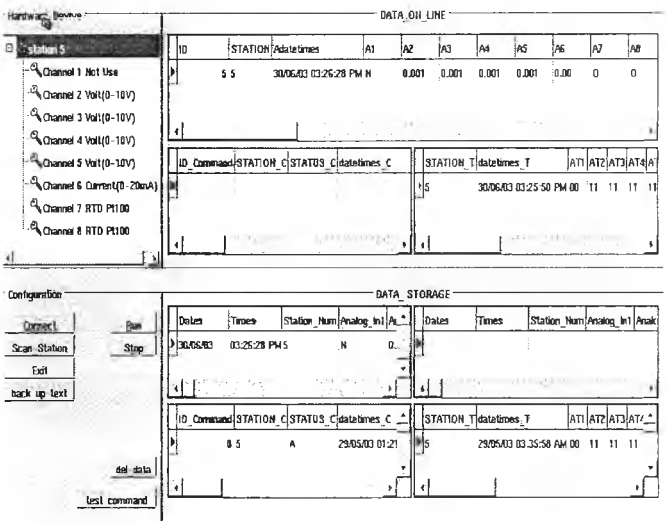
รูปที่ 6 ลักษณะของข้อมูล Database Temporary ของ Linux Server

2. Permanent Database จะทำหน้าที่ในการบันทึกข้อมูลทุกประเภทไว้ในฐานข้อมูลประวัติโดยจะทำการบันทึกข้อมูลไปพร้อมกับฐานข้อมูลประเภทชั่วคราว (Temporary Database) และจะมีการสำรองข้อมูลในรูปของไฟล์เอกสารตามเวลาที่กำหนดไว้ โดยหน้าที่หลักของฐานข้อมูลชนิดนี้คือช่วยในการดูประวัติย้อนหลังในการทำงานและสนับสนุนการทำรายงานรวมทั้งการป้องกันข้อมูลสูญหายถ้าระบบฐานข้อมูลประเภทชั่วคราวเกิดการขัดข้อง

4.2 โครงสร้างส่วนติดต่อกับผู้ใช้ (User Interface)

เราจะสามารถแบ่งได้เป็น 2 ส่วนใหญ่คือในฝั่งลูกข่าย (Client)กับตัวให้บริการ (Server) โดยที่ในตัวฝั่งให้บริการอาจมีส่วนลูกข่ายด้วยก็ได้

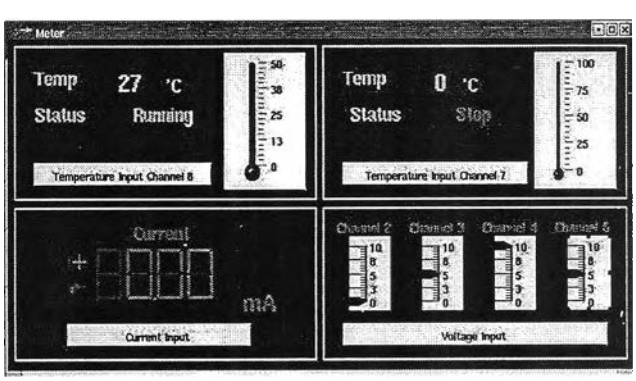
4.2.1 ส่วนติดต่อกับผู้ใช้ของตัวให้บริการ จะทำหน้าที่ในการตรวจสอบการติดต่อและการกำหนดค่าต่างๆให้หน่วยควบคุมปลายทางระยะไกลรวมทั้งแสดงสถานะความเปลี่ยนแปลงทุกอย่างที่เกิดขึ้นขณะบันทึกข้อมูลมูลลงฐานข้อมูลดังอธิบายไว้ใน โครงสร้างแรก



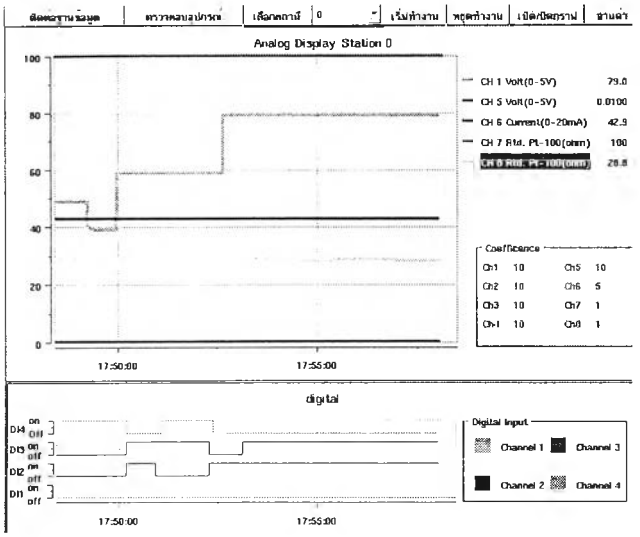
รูปที่ 7 ส่วนติดต่อกับผู้ใช้ของตัวให้บริการ

4.2.2 ส่วนติดต่อกับผู้ใช้ของลูกข่าย ได้กำหนดให้มีคุณสมบัติดังต่อไปนี้

แสดงผลการทำงานในขณะเวลาจริง (Real - Time Monitoring) สามารถตรวจสอบตามการทำงานของระบบสกาดาในรูปแบบของตัวอักษร กราฟ หรือกราฟฟิกอื่นๆที่สามารถสื่อความหมายได้อย่างชัดเจนในขณะที่เทียบกับเวลาจริง ดังแสดงในรูปที่ 8 และ รูปที่ 9

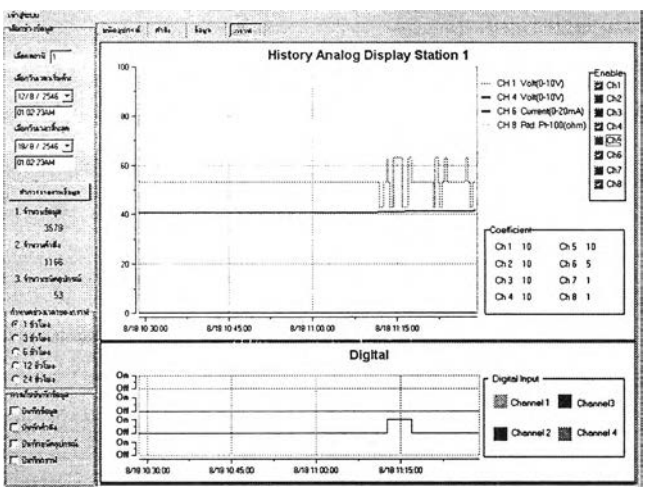


รูปที่ 8 การวัดในรูปมาตรวัด



รูปที่ 9 การวัดในแบบของรูปกราฟ

แสดงผลประวัติการทำงานในรูปแบบเวลาจริง (Real-Time and Historical Trends) สามารถดูการเปลี่ยนแปลงของค่าต่างๆเทียบกับเวลาได้ ดังแสดงในรูปที่ 10



รูปที่ 10 ประวัติการทำงานในรูปแบบของกราฟ

สามารถกำหนดเกณฑ์ในแจ้งความผิดพลาดของค่าข้อมูลที่รับเข้ามา (Alarm Capabilities) ในรูปแบบสัญญาณเตือนที่เกิดขึ้นสามารถแสดงบนหน้าจอในลักษณะการเปลี่ยนแปลงของสีได้

ระบบรับ - ให้บริการ (Client-Server System) ในระบบที่ทดลองสร้างขึ้นมานั้น ระบบควบคุมสามารถทำงานบนระบบเครือข่าย ในรูปแบบระบบรับ-ให้บริการ (Client -Server) ทำให้สามารถกระจายงานต่างๆ ไปยังคอมพิวเตอร์แต่ละตัวในเครือข่าย โดยอาศัยข้อมูลกลางร่วมกันได้ทั้งระบบปฏิบัติการที่เป็น ลีนุกซ์หรือไมโครซอฟต์วินโดว์

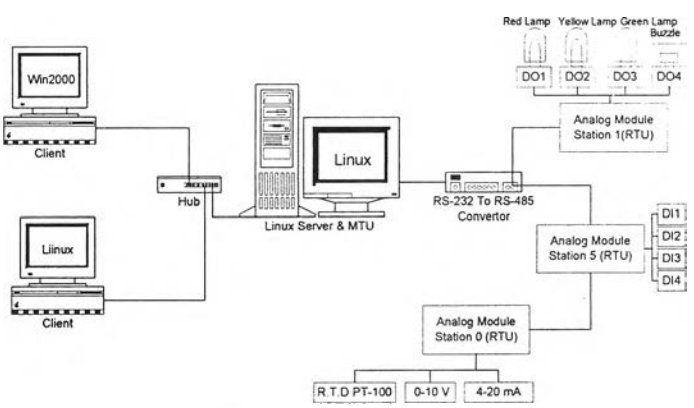
สนับสนุนการทำรายงาน (Reporting) รายงานประจำวันในรูปของไฟล์เอกสารเกี่ยวกับข้อมูลต่างๆ เพื่อเป็นข้อมูลสำหรับใช้ในการตัดสินใจ

สรุปผล

จากการทดลองพัฒนาระบบสกาดาบนระบบปฏิบัติการลีนุกซ์ในห้องปฏิบัติการตามข้อกำหนดดังไว้ที่หัวข้อการออกแบบลีนุกซ์สกาดา (3) โดยระบบที่ได้มีคุณสมบัติที่ไม่รวมความสามารถทาง User Interface ของส่วนโปรแกรมสกาดา มีดังต่อไปนี้

- สถานีหลัก (MTU) สามารถติดต่อกับหน่วยควบคุมปลายทางระยะไกล (RTU) ด้วยโพรโตคอลสื่อสารแบบ MOBUS ASCII บนมาตรฐาน RS-485 ได้ 10 หน่วย และสามารถเพิ่มได้ถึง 32 หน่วย
- ความเร็วสูงสุดที่สถานีหลักอ่านข้อมูลจากหน่วยควบคุมปลายทางระยะไกล 1 หน่วย เท่ากับ 1 วินาที
- สามารถตรวจความผิดปกติจากการสื่อสารระหว่างสถานีหลักอ่านข้อมูลจากหน่วยควบคุมปลายทางระยะไกลระหว่างการทำงานได้ เช่นกรณีที่สายส่งสัญญาณหลุด
- สนับสนุนการควบคุมระยะไกลจากเครื่องลูกข่าย (Client)

เพื่อสนับสนุนการทำงานของโปรแกรมและระบบสกาดาที่ออกแบบไว้ ได้มีการสร้างระบบฮาร์ดแวร์ขึ้นมาทดสอบการทำงาน ดังแสดงในรูปที่ 11



รูปที่ 11 โครงสร้างการติดตั้งระบบทดสอบการทำงานของระบบควบคุมและรวบรวมข้อมูลสกาดาบนระบบปฏิบัติการลีนุกซ์

สรุปผลการทดสอบ

5.1 สามารถพัฒนาโปรแกรมที่ใช้งานในระบบสกาดาบนคอมพิวเตอร์ที่มีลีนุกซ์เป็นระบบปฏิบัติการโดยสถานีหลัก (MTU) เป็นแบบรวมฟังก์ชันไว้ที่ศูนย์กลาง และมีฟังก์ชันการทำงานเช่น การแสดงผลการทำงานใน

ขณะเวลาจริง (Real - Time Monitoring) ในรูปแบบที่สื่อความหมายได้อย่างชัดเจน

5.2 สามารถติดต่อกับสถานีปลายทางระยะไกล (RTU) ซึ่งมีช่องทางรับ / ส่งสัญญาณ ตามมาตรฐาน RS-485 ในรูปแบบของสัญญาณแอนะล็อก (Analog Signal) และสัญญาณดิจิทัล (Digital Signal) ระหว่างสถานีหลักกับสถานีปลายทางระยะไกลจำนวน 10 สถานีได้อย่างมีประสิทธิภาพ รวมถึงสามารถตรวจจับความผิดปกติจากการสื่อสารระหว่างกันได้

5.3 สามารถให้บริการเครื่องลูกข่ายที่ใช้ระบบปฏิบัติการลีนุกซ์หรือไมโครซอฟต์วินโดว์ ได้ในรูปแบบของ ลูกข่าย-ผู้ให้บริการ (Client-Server) รวมถึงสามารถควบคุมการทำงานระยะไกลจากเครื่องลูกข่าย (Client) มายังที่สถานีปลายทางระยะไกลโดยผ่านสถานีหลักได้

จากการพัฒนาระบบสกาดาบนระบบปฏิบัติการลีนุกซ์ ผู้พัฒนาสามารถสรุปถึงจุดเด่นในการนำระบบปฏิบัติการลีนุกซ์มาใช้งานในระบบสกาดาได้ดังต่อไปนี้

1. ความคล่องตัวในการใช้ฮาร์ดแวร์มีสูง อย่างเช่นลีนุกซ์จะมองอุปกรณ์ฮาร์ดแวร์เป็นไฟล์ (เหมือนไฟล์เอกสาร) ดังนั้นการทำงานกับอุปกรณ์จึงมีความคล่องตัวและยืดหยุ่นสูงในการนำไปใช้งาน

2. การถ่ายโอนข้อมูล ลีนุกซ์มีความสามารถในการใช้ ไฟล์ร่วมกับระบบปฏิบัติการอื่นๆได้ เช่น FAT32 ของ WIN98, NTFS ของ WINNT เป็นต้น จึงทำให้สะดวกต่อการถ่ายโอนข้อมูล

3. ระบบเครือข่าย ลีนุกซ์สนับสนุนการทำงานกับระบบเครือข่าย (Network System) บน TCP/IP ได้อย่างสมบูรณ์แบบ

4. เครื่องมือที่ใช้พัฒนา สามารถหาได้ง่ายทั้งในรูปแบบทางการค้าและของฟรีอย่างเช่น Borland Kylix [6] เป็นโปรแกรมเชิงภาษาภาพ(Visual Programming for Linux) ที่ทำงานคล้ายกับ Delphi , Borland C++, VB บนไมโครซอฟต์วินโดว์ ทำให้ง่ายและสะดวกและประหยัดเวลาต่อการ พัฒนา

5. มีผู้ร่วมพัฒนาอยู่ทั่วโลก มีการแลกเปลี่ยนข้อมูลตลอดเวลาถึงข้อดีและข้อเสียของระบบปฏิบัติการและคำแนะนำในการแก้ปัญหา

6. ระบบปฏิบัติการที่เปิด (Open Source) สิ่งที่สำคัญที่สุดคือลีนุกซ์เป็นระบบปฏิบัติการที่เปิด (Open Source) ทำให้เราสามารถแก้ไขและเปลี่ยนแปลงทุกอย่างของรหัสต้นฉบับได้ตามความต้องการ

เอกสารอ้างอิง

[1] IEEE recommended practice for master/remote supervisory control and data acquisition (SCADA) communications ฉบับภาษาไทย, แปลและเรียบเรียงโดย สมาคมสถาบันวิศวกรไฟฟ้าและอิเล็กส์ทรอนิกส์แห่งประเทศไทย (IEEE Thailand Section) , กรุงเทพฯ : การไฟ ฟ้าส่วนภูมิภาค, 2539

[2] IEEE Standard Definition , Specification and Analysis of Systems Used For Supervisory Control , Data Acquisition and Automatic Control (IEEE Std. 37.1 - 1994)

[3] Chen Qizhi; Qian Qinquan;,"The research of UNIX platform for SCADA",Power Engineering Society Winter Meeting, 2000. IEEE pp. 2041-2045 vol.3, 23-27 Jan. 2000

[4] AEG SHNIDER AUTOMATION. "Modicon Modbus Network Planning and Installation Guide. Version 3.0", 1996, 890 USE 10000. U.S.A

[5] Unel G.; Ambrosini G.; Conka T.; Crone G.; Fernanedes A.; Francis D.; Joos M.; Lehmann G.; Lopez J.; Mailov A.; Mapelli L.; Mornacchi G.; Niculescu M.; Petersen J.; Tremblet L.; Veneziano S.; Wildish T.; Yasu Y.;, "Using Linux PCs in DAQ applications ", Real Time Conference, 1999. Santa Fe 1999. 11th IEEE NPSS, pp. 73 – 77, 14-18 June 1999

[6] "Kylix 3 Enterprise Trial and Companion Tools For Kylix" , http://www.borland.com, http://sourceforge.net

[7] "Modbus Protocol", http://www.modicon.com/techpubs/intr7.html

[8] "MySQL Database Server", http://www.mysql.com

ชนวรรณ เกตุดี สำเร็จการศึกษาระดับปริญญาตรี สาขาวิศวกรรมไฟฟ้าจากมหาวิทยาลัยเกษตรศาสตร์ ในปี พ.ศ.2542 ปัจจุบันกำลังศึกษาระดับปริญญาโท ที่จุฬาลงกรณ์มหาวิทยาลัย

กฤษดา วิศวธีรานนท์ สำเร็จการศึกษาระดับปริญญาตรี และ โท สาขาวิศวกรรมไฟฟ้าจากมหาวิทยาลัยเกียวโต ประเทศญี่ปุ่น ปัจจุบันดำรงตำแหน่งรองศาสตราจารย์ ประจำภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

มานพ วงศ์สายสุวรรณ สำเร็จการศึกษาระดับปริญญาตรี สาขาวิศวกรรมไฟฟ้าจากมหาวิทยาลัยจุฬาลงกรณ์มหาวิทยาลัยในปี พ.ศ.2531 ระดับปริญญาโทและเอก สาขาวิศวกรรมระบบควบคุมจาก Tokyo Institute of Technology เมื่อปี พ.ศ. 2534 และ 2537 ตามลำดับปัจจุบันดำรงตำแหน่งผู้ช่วยศาสตราจารย์ ประจำภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ประวัติผู้เขียนวิทยานิพนธ์

นายชนวรรณ เกตุดี เกิดวันที่ 18 ตุลาคม พ.ศ. 2519 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมไฟฟ้า จากมหาวิทยาลัยเกษตรศาสตร์ เมื่อปี พ.ศ. 2542 เคยทำงานในตำแหน่งวิศวกรไฟฟ้าที่บริษัทออโตอัลไลน์แอน (Auto Alliance Thailand) จำกัด เป็นเวลา 1 ปี ก่อนออกมาศึกษาในหลักสูตรวิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมไฟฟ้าที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ 2544