Bangladesh University of Engineering & Technology

A report on

IMPLEMENTATION OF DATA ACQUISITION AND OVERCURRENT PROTECTION FOR SINGLE PHASE SYSTEM

Prepared for:

EEE 478: Power System Protection Laboratory

Dr. Shahidul Islam khan *Professor* *Department of EEE* *BUET.*	*Imtiaz Ahmed* *Lecturer* *Department of EEE* *BUET.*

Prepared by:

Group No. 05

Name	Student No.
Abul Hasan Fahad	0806021
Md. Mehedi Hasan	0806023
Pertha Protim Dutta	0806024
Md. Awal	0806043
Md. Zabir Ahmed	0806054
Zadid Khan	0806056
IK Shimul	0806126

30 March, 2014

Objective:

The objectives of this project are-

(1) To implement a data acquisition, display and instantaneous over current protection system for single phase load.
(2) To calculate Instantaneous Vrms, Irms, Real Power (Watts), Apparent power (VA) and power factor(pf) of the load

Formulae:

Formulas for calculating various power system state variables from sampled data are given below:

(1) $V_{rms}= \sqrt{\frac{(\sum_{i=1}^{N} v(i)^2)}{N}}$ (volts)

(2) $I_{rms}= \sqrt{\frac{(\sum_{i=1}^{N} I(i)^2)}{N}}$ (amps)

(3) $P= \frac{(\sum_{i=1}^{N} v(i) * I(i))}{N}$ (watts)

(4) $S= V_{rms} * I_{rms}$ (Volt-amps)

(5) $Pf = \frac{P}{S} = \frac{(\sum_{i=1}^{N} v(i) * I(i))/N}{Vrms * Irms}$

Algorithm:

Step-1: Capture actual system data (ac) by CT and PT

Step-2: Signal conditioning of CT and PT output for sampling by microcontroller (addition of dc offset, signal retains original shape)

Step-3: Sample the current and voltage data at a frequency higher than 100 Hz by the microcontroller

Step-4: Continuously Serial Transmit the data to the computer from microcontroller

Step-5: Capture the data samples from microcontroller by the main program in the computer as a set of data and continuously plot the captured data-window

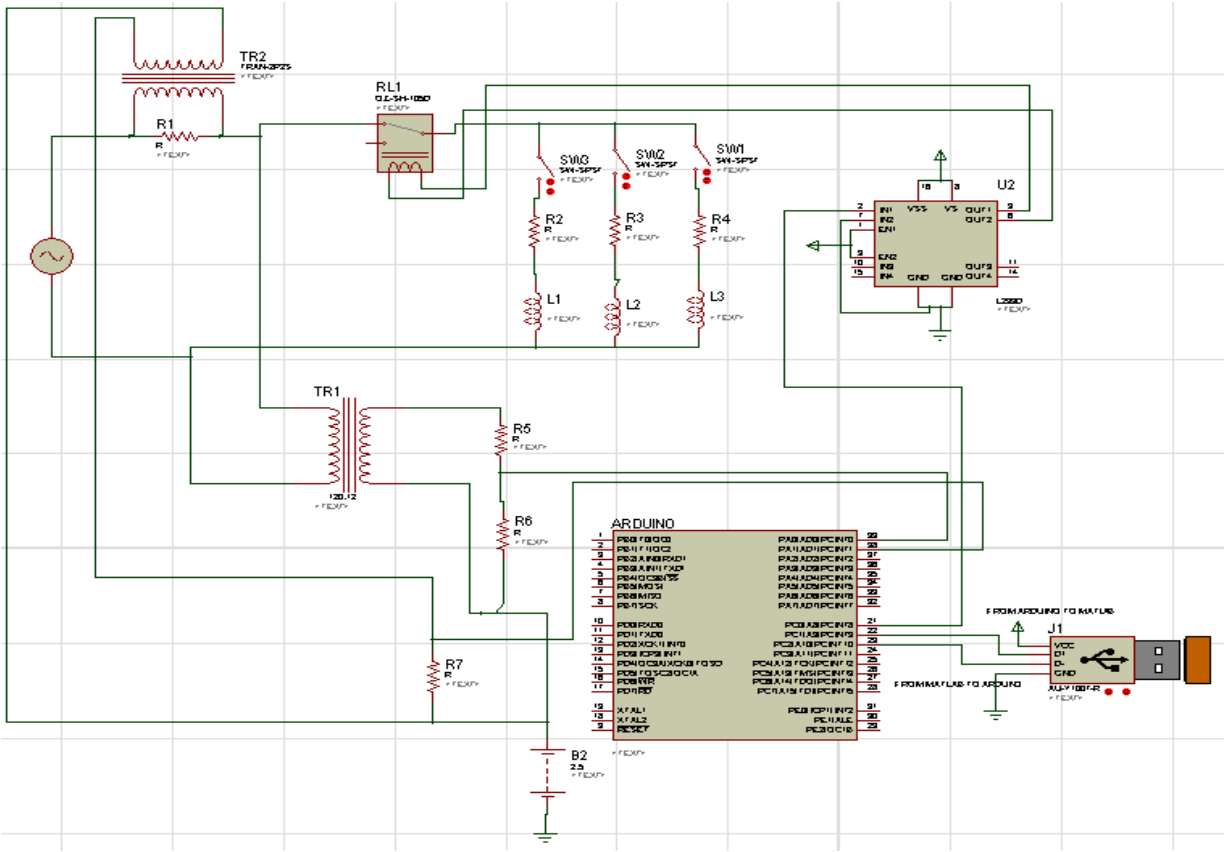
Step-6: Perform calculation and analysis on the real time data and show present state of the system

Step-7: Goto step-5 for a new set of data samples

Circuit Components:

1) Potential transformer
2) Current transformer
3) Microcontroller board (arduino)
4) Bulb(load)
5) Relay (attracted armature type)
6) Relay driver IC
7) Resistors
8) Capacitor

Circuit Diagram:



Implementation of Data Acquisition:

Voltage data Acquisition:

The loads are connected in series with the source through a relay. The voltage is first stepped down using a 220: 12 transformer. The output voltage of the transformer is divided using a voltage divider circuit, thus Peak output voltage is 2.5 volt in normal condition. As microcontrollers can't work with negative voltage, a dc offset of 2.5 volt is added to make it within 0 volt to 5 volt range, before feeding it to the Arduino input.

Current data Acquisition:

The load current is measured using a current transformer (CT). The output of the CT is connected to a 1 KΩ resistance. So, a voltage proportional to the load current is generated across this resistance. A dc offset of 2.5 volt is also added with this voltage, making it perfect for sampling.

Voltage & Current Sampling:

The voltage and current is sampled using the analog to digital conversion (ADC) feature of the Arduino. In this feature, the sampling data is represented in 10 bit binary numbers. So, the highest value of data (in decimal) will be 1024 and the lowest will be zero.

Data Transmission:

We need to send voltage samples and current samples simultaneously. To do this, the voltage data is multiplied with 10000. So, the data is shifted to left by 5 digits. Then we added the current data with it. This encoded data which has both the voltage and current information is sent to matlab using serial communication.

Data decoding and display:

The data received in the matlab is decoded just by dividing it by 100000. The ratio is the voltage data and the residue is the current data. These data is converted to the real value by scaling it by 5/1024 and then subtracting the dc offset from it. 200 data points are acquired at a time in a window and is displayed.

Estimation of system parameters:

- *Frequency:* We used correlation property of a signal to determine the frequency. We know that if we determine autocorrelation coefficients of a signal, we will find high peak values after each period. We calculated the points between two subsequent peaks in the correlation coefficients and measured the period of signal using the time for obtaining that number of data points.

- *RMS quantities*: Each sample in the window is squared and then all of them are summed. Then they are divided by the window-length. Taking the squre root of this quantity leads us to the Root Mean Sqaure Value of the data set. In this way, voltage and current RMS values have been calculated. (Formula 1 and 2 of first page of this report).

- *Power:* Real power is calculated by multiplying voltage and corresponding current samples of each window and then their sum is divided by the window length. The expression can be found on formula (3) of first page. Apparent power has been calculated using the values of Vrms and Irms. Formula no.4 is in the first page of the report.

- *Power Factor:* It is simply the ration of Real Power to Apparent Power.

Codes have been attached at the end of the report.

Implementation of Overcurrent Protection:

In matlab we are continuously calculating the rms value of voltage and current. The load is connected with the source through a relay. If the current becomes higher than a predetermined value, a trip signal is sent to arduino by the matlab. The arduino then makes the relay trip, thus isolating the overload from the system.

The relay needs considerable amount of current to attract the armature of its structure. Hence the relay is energized by a ULN 2003. ULN2003 is a high current driver IC. The input of the ULN2003 is connected to the arduino. In normal condition, the input pin is kept high so the relay stays closed. The trip signal from microcontroller makes the input pin of ULN2003 low which in turns trips the relay and disconnects the load from the line.

After a certain time the relay is energized to make the circuit and the rms value of the instantaneous current is checked. If the value is still higher than the limit than the relay again breaks the circuit. Otherwise, the normal operation of the system continues.

Components and Cost:

Sl No	Components	Number	Cost (BDT)
1	A 220:12 volt step down transformer	1	80
2	Resistors (470kΩ, 33kΩ, 1kΩ, 10kΩ)	-	10
3	Capacitor	1	5
4	6V Relay	1	25
5	Current Transformer	1	100
6	ULN 2003		150
7	Arduino Board	1	2000
8	Connectors	-	25
9	Load (100 watt bulb)	3	150
10	PCB	1	250

Total = 2795 BDT

Applications for our project:

1. Measurement of instantaneous voltage and current of a circuit
2. Observation of shapes of voltage and current
3. Measurement of system frequency
4. Measurement of power factor
5. Determination of whether power factor improvement is necessary or not.
6. Determination of whether there is any disturbance in the system voltage.
7. To provide over current protection to a system

Hardware Implementation:

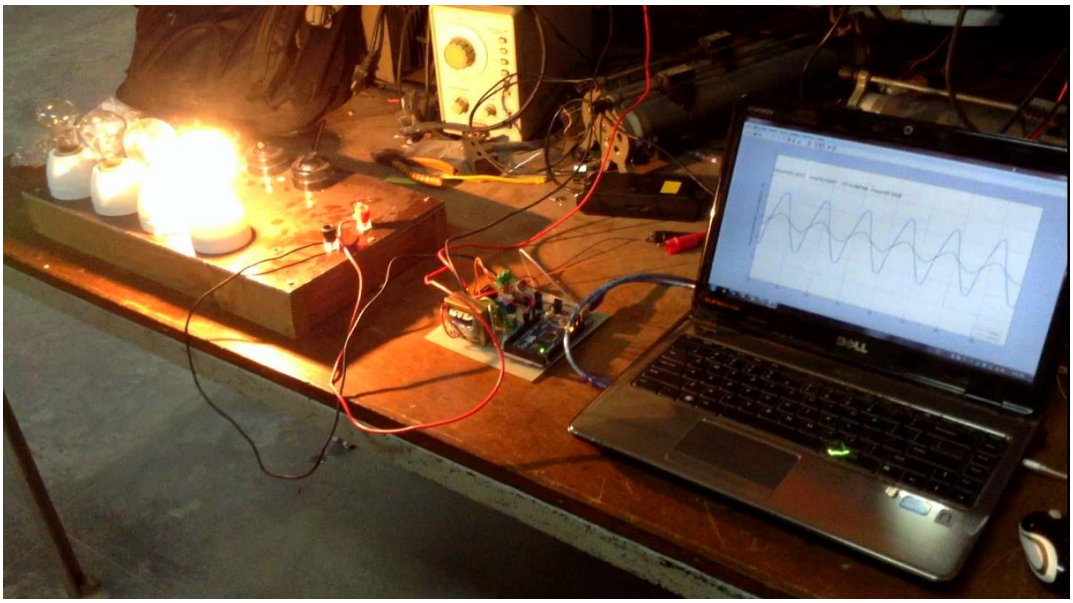


Figure: Hardware assembly of our project, interfaced with Matlab

Conclusion:

Our project has provided us with practical understanding of data acquisition and supervisory control. We have also gained valuable insights of microprocessor-based overcurrent protection through our implementation. Certainly, further improvements of our project will enable us to touch other aspects of power system protection and control arena. Lastly, we want to express utmost respect to our teachers who have provided us with the opportunity to explore such a fascinating project idea.

Codes:

Arduino Codes:

```
const int analogInPin = A0; // Analog input pin that the potentiometer is attached to

const int analogInPin2 = A1;

const int outpin =9;

long sensorValue = 0;   // value read from the pot

long sensorValue2=0;

long summ=0;

char mat='n';

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(115200);
  pinMode(outpin,OUTPUT);
  analogWrite(outpin,128);}
void loop() {
  // read the analog in value:s
  sensorValue = analogRead(analogInPin);
  sensorValue2 = analogRead(analogInPin2);
  summ=sensorValue*10000+sensorValue2;
  // map it to the range of the analog out:

  // print the results to the serial monitor:
  Serial.println((summ));



  if (Serial.available()>0)
  mat=Serial.read();
  end

  // wait 2 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
 // delay(1);
}
```

Matlab Code:

```
clear all
close all
clc

s=serial('com9');
s.baudrate=115200;
% set(s,'inputbuffersize',2);
fopen(s);
ref=2.5;
vmul=257;
imul=1.93;
v_lim=0;
c_lim=1;
wait_time=5;
window=1:200;
windowsize=length(window)-1;
data=zeros(1,windowsize+1);
datav=zeros(1,windowsize+1);
datai=zeros(1,windowsize+1);
idx=1;
pf=0;
Irms=0;
Vrms=0;
Vmax=0;
Imax=0;
t_flag=0;
fs=1/100e-6;
plothandle=plot(datav);
while(ishandle(plothandle))
    tic
    t_now=clock;

    if t_flag
        if etime(t_now,t_trip)>wait_time
            fprintf(s,'U');   % Command to reconnect load
            t_flag=0;
        end
    end

    for idx=1:windowsize
        if s.bytesavailable > 0
            adc=fscanf(s,'%u');
            %             s.bytesavailable=0;
            % adc=fgetl(s);
            if length(adc)==1 && idx>1
                data(idx-1)=adc(end);

            end
        end
    end
    tt=toc;
    datai=mod(data,10000);
    datai=datai*5/1024-ref;
    datav=round(data/10000);
```

```
    datav=datav*5/1024-ref;


    Vmax=max(datav);Imax=max(datai);

    VV=(datav(1:end-1).*datav(2:end));
    II=(datai(1:end-1).*datai(2:end));

    indv=find(VV<0);
    indi=find(II<0) ;
    Vrms=0;
    Irms=0;
    rangev=0;rangei=0;
    if length(indv)>3

        rangev=(indv(end-1)-indv(end-3)+1);
        %    range=26;
        Vrms=sqrt(sumsqr(datav(indv(end-3):indv(end-3)+rangev-1))/rangev);
% calculate Vrms
    end
    if length(indi)>3

        rangei=(indi(end-1)-indi(end-3)+1);
        %    range=26;
        Irms=sqrt(sumsqr(datai(indi(end-3):indi(end-3)+rangei-1))/rangei);
% calculate Irms
    end
    %    Irms=sqrt(sumsqr(datai)/(windowsize));
    %         Irms=sqrt(sumsqr(datai(indi(end-3):indi(end-3)+range-
1))/range);
    range=min(rangev,rangei);
    P=0;
    if length(indv)>3 && length(indi)>3
        P=sum(datav(indv(end-3):indv(end-3)+range-1).*datai(indi(end-
3):indi(end-3)+range-1))/range;  %calculate real power
    end
    S=Vrms*Irms; % calculate apparent power
    pf=P/S;  %calculate power factor



    if Irms*imul>c_lim
        fprintf(s,'T');  % Command to disconnect the load
        t_trip=clock;
        t_flag=1;
        %         1
    end

    %    P
    acor=xcorr(datav);
    [peaks,tlocs]=findpeaks(acor);
    T_per=diff(tlocs); T_sample=median(T_per);
    T_sec=T_sample;
    fs=windowsize/tt*2;
    f=1/T_sec*fs;  % calculate frequency
```

```
    if(~isnan(pf))

        str=['Vrms=' num2str(vmul*Vrms)  '    Irms=' num2str(Irms*imul), '
PF=' num2str(pf)  '   Freq=' num2str(f)];
    else
        str=['Vrms=' num2str(vmul*Vrms)  '    Irms=' num2str(Irms*imul), '
PF=N/A (LOAD DISCONNECTED)'   '   Freq=' num2str(f)];
    end
    %     dispstr={'Vrms=' num2str(vmul*Vrms), '    Irms='
num2str(Irms*imul), ...
    %         '    PF=' num2str(pf) ,'   Freq=' num2str(f)};
    figure(1)
    plothandle=plot(window,datav,window,datai,'Linewidth',2);ylim([-3
3]);xlim([40 windowsize-1]);
    grid on
    ylabel(['Voltage Multiplier= ' num2str(vmul) '  Current Multiplier= '
num2str(imul)]);
    %     subplot(121),plot(datav(2:end),'linewidth',2);ylim([0 6]);grid on
    %     subplot(122),plot(datai(2:end),'linewidth',2);ylim([0 6]);grid on
    %     pause(.2);
    text(40,2.25,str,'Fontsize',14);
    %
annotation('textbox',[.2,.4,.1,.1],'String',dispstr,'FitBoxToText','on');

    legend('Voltage','Current','Location','Best');
    flushinput(s);
    drawnow
end


%%
fclose(s)
```