Appendix II: Smart Socket Source Code

```
//Sketch senses if there is any current fllow.
//and then calculates useful values like real power, Irms.

#include <WString.h>
#include <Ethernet.h>
#include <avr/interrupt.h>

//******************************************************************************
********************************
//Setup variables
int numberOfSamples = 3000;
//******************************************************************************
********************************

//******************************************************************************
********************************
//Set Status,Toggle, Voltage, and current input pins
int inPinV  = 2;
int inPinI  = 1;
int outPinSwitch = 4 ;
int inPinToggle  = 2;

//int outPinCS=0, inPinTS=0,inPinTSOld=0;
//******************************************************************************
********************************

//******************************************************************************
********************************
//Calibration coeficients
//These need to be set in order to obtain accurate results
double VCAL = 1.0;
double ICAL = 1.0;
double PHASECAL = 2.3;
//******************************************************************************
********************************

//******************************************************************************
********************************
//Sample variables
int lastSampleV,lastSampleI,sampleV,sampleI;
//******************************************************************************
********************************

//******************************************************************************
********************************
```

```
//Filter variables
double lastFilteredV, lastFilteredI, filteredV, filteredI;
double filterTemp;
//********************************************************************************
*********************************

//********************************************************************************
*********************************
//Stores the phase calibrated instantaneous voltage.
double calibratedV;
//********************************************************************************
*********************************

//********************************************************************************
*********************************
//Power calculation variables
double sqI,sqV,instP,sumI,sumV,sumP;
//********************************************************************************
*********************************

//********************************************************************************
*********************************
//Useful value variables
double realPower, Vrms, Irms;
boolean flag;
//********************************************************************************
*********************************

//********************************************************************************
*********************************
//Ethernet Variables
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFD, 0xEF };
byte ip[] = { 10, 0, 0, 51 }; // ip in lan
byte gateway[] = { 10, 0, 0, 2 }; // internet access via router
byte subnet[] = { 255, 0, 0, 0 }; //subnet mask
Server server(80); //server port
String readString = String(30); //string for fetching data from address
//********************************************************************************
*********************************

//**************************************Loop Function Starts
here************************************************
void setup()
{
  //start Ethernet
  Ethernet.begin(mac, ip, gateway, subnet);
  //enable serial datada print
  Serial.begin(9600);
  setPinMod();
```

```
}

void loop()
{

  //calculatePower();
  listenHttpRequest();
// Create a client connection

}
//************************************Loop Function Ends
here*********************************************



//************************************listenHttpRequest Function Starts
here*******************************
void listenHttpRequest(){

  Client client = server.available();
  if (client) {
    if (client.connected()) {
      if (client.available()) {
        char c = client.read();
        //read char by char HTTP request
        if (readString.length() < 30) {
          //store characters to string
          readString.append(c);
        }

        //output chars to serial port
        Serial.print(c);
        //if HTTP request has ended
        if (c == '\n') {
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println();
          //set background to yellow
          client.print("<body style=background-color:yellow>");
          //send first heading
          client.println("<font color='red'><h1>The Appliance Status at IP:
192.168.1.235</font></h1>");
          client.println("<hr />");

          if(readString.contains("S=0")) {
              //led has to be turned OFF
              chnageApplianceStatus(false);
              client.println("<h1>The Status is Need to be set to OFF</h1>");
            }
```

```
     if(readString.contains("S=1")) {
      chnageApplianceStatus(true);
          client.println("<h1>The Status is Need to be set to ON</h1>");
         }

     if(readString.contains("S=2")) {
     client.println("<h1>The Appliance Power is: </h1>");
     realPower=0;
     calculatePower();
     calculatePower();
     calculatePower();
     calculatePower();
     //calculatePower();
     //calculatePower();
     client.println(realPower,DEC);

     if (realPower>= 0.10)
      client.println("<h1>The Appliance is Powered ON </h1>");
     else
      client.println("<h1>The Appliance is Powered OFF</h1>");

     }

     client.println("<hr />");
     client.println("</body></html>");
     //clearing string for next read
     //stopping client
     client.stop();

     readString="";
     delay(10);
    }
   }
  }
 }
}
//***********************************listenHttpRequest Function Ends
here***********************************


//***********************************setPinMod Function Starts
here****************************************
void setPinMod()
{
 pinMode(inPinV, INPUT);
 pinMode(inPinI, INPUT);
 pinMode(inPinToggle, INPUT);
 pinMode(outPinSwitch, OUTPUT);
}
//*************************************setPinMod Function End
here*******************************************



//*************************************chnageApplianceStatus Function Starts
here****************************
void chnageApplianceStatus(boolean stat)
{
 if(stat==true)
 {
  digitalWrite(outPinSwitch, HIGH);

 }
 else
 {
  digitalWrite(outPinSwitch, LOW);

 }
}
//*************************************chnageApplianceStatus Function Ends
here******************************


//*************************************calculatePower Function Starts
here***********************************
void calculatePower()
{
 for (int n=0; n<numberOfSamples; n++)
 {

  //Used for offset removal
  lastSampleV=sampleV;
  lastSampleI=sampleI;

  //Read in voltage and current samples.
  sampleV = analogRead(inPinV);
  sampleI = analogRead(inPinI);

  //Used for offset removal
  lastFilteredV = filteredV;
  lastFilteredI = filteredI;

  //Digital high pass filters to remove 2.5V DC offset.
  filteredV = 0.996*(lastFilteredV+sampleV-lastSampleV);
  filteredI = 0.996*(lastFilteredI+sampleI-lastSampleI);
```

```
    //Phase calibration goes here.
    calibratedV = lastFilteredV + PHASECAL * (filteredV - lastFilteredV);

    //Root-mean-square method voltage
    //1) square voltage values
    sqV= calibratedV * calibratedV;
    //2) sum
    sumV += sqV;

    //Root-mean-square method current
    //1) square current values
    sqI = filteredI * filteredI;
    //2) sum
    sumI += sqI;

    //Instantaneous Power
    instP = calibratedV * filteredI;
    //Sum
    sumP +=instP;
  }//For Loop Ends here

  //Calculation of the root of the mean of the voltage and current squared (rms)
  //Calibration coeficients applied.
  //Vrms = VCAL*sqrt(sumV / numberOfSamples);
  Irms = ICAL*sqrt(sumI / numberOfSamples);

  //Calculation power values
  realPower = VCAL*ICAL*sumP / numberOfSamples;
  //apparentPower = Vrms * Irms;
  //powerFactor = realPower / apparentPower;

  //Output to serial
  Serial.print(realPower);
  Serial.print(' ');
  Serial.println(Irms);

  if(realPower>0.05)
    flag=true;
  else
    flag=false;

  //Reset accumulators
  sumV = 0;
  sumI = 0;
  sumP = 0;
}
//************************************calculatePower Function Ends
here***************************************
```

Appendix III: Web Service Source Code

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.Services;
using System.Data;
using System.Data.SqlClient;
using Npgsql;
namespace DSMWebService

{

/// <summary>

/// Summary description for Service1

/// </summary>

 [WebService(Namespace = "http://convergence.ac.za")]

 [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]

 [System.ComponentModel.ToolboxItem(false)]

// To allow this Web Service to be called from script, using ASP.NET AJAX,
uncomment the following line.

// [System.Web.Script.Services.ScriptService]

public class DSMService : System.Web.Services.WebService
{

[WebMethod]
private int getCurrentStatePG(string ipAddress)
{
return 2;
}

[WebMethod]
public int postLogEntryAndGetStatusPG(int userId, string ipAddress, string
reading)
{
NpgsqlConnection con = new NpgsqlConnection();
con.ConnectionString =
"server=localhost;database=MehrozeDSM;uid=postgres;pwd=santana";
NpgsqlCommand cmd = new NpgsqlCommand();
cmd.Connection = con;
int log_id = getMaxIdPG("tbl_appliance_reading_log", "log_id");
```

```
cmd.CommandText = "INSERT INTO tbl_appliance_reading_log(log_id, user_id,
date_time, ip_address, reading) VALUES( " + log_id + "," + userId + ", '" +
DateTime.Now.ToString() + "', '" + ipAddress + "', " + reading + ")";
con.Open();
cmd.ExecuteNonQuery();
con.Close();
//Get the Appliance Status and return the response to the Appliance
return getCurrentStatePG(ipAddress);
}
private int getMaxIdPG(string tableName, string colName)
{
NpgsqlConnection con = new NpgsqlConnection();
con.ConnectionString =
"server=localhost;database=MehrozeDSM;uid=postgres;pwd=santana";
NpgsqlDataReader dr;
NpgsqlCommand cmd = new NpgsqlCommand();
cmd.Connection = con;
cmd.CommandText = "Select max(" + colName + ")+1 from " + tableName;
con.Open();
dr = cmd.ExecuteReader();
dr.Read();
int maxId = (int)dr[0];
dr.Close();
con.Close();
return (maxId);

}

//*******************MS SQL SERVER STARTS FROM
HERE************************************************************

 [WebMethod]
private int getCurrentStateMSSQL(string ipAddress)
{
SqlConnection con = new SqlConnection();
con.ConnectionString =
"server=localhost;database=MehrozeDSM;uid=sa;pwd=santana";
SqlDataReader dr;
SqlCommand cmd = new SqlCommand();
cmd.Connection = con;
cmd.CommandText = "Select * from tbl_appliance_status where
status_datetime=(select max (status_datetime) from tbl_appliance_status)";
con.Open();
dr = cmd.ExecuteReader();
dr.Read();
int status=(int)dr[2];
dr.Close();
con.Close();
return (status);
```

```
}

[WebMethod]

public int postLogEntryAndGetStatusMSSQL(int userId, string ipAddress, string
reading)
{
SqlConnection con = new SqlConnection();
con.ConnectionString =
"server=localhost;database=MehrozeDSM;uid=sa;pwd=santana";
SqlCommand cmd = new SqlCommand();
cmd.Connection = con;
int log_id = getMaxIdMSSQL("tbl_appliance_reading_log", "log_id");
cmd.CommandText = "INSERT INTO tbl_appliance_reading_log(log_id, user_id,
date_time, ip_address, reading) VALUES( " + log_id + "," + userId + ", '" +
DateTime.Now.ToString() + "', '" + ipAddress + "', " + reading + ")";
con.Open();
cmd.ExecuteNonQuery();
con.Close();
//Get the Appliance Status and return the response to the Appliance
return getCurrentStateMSSQL(ipAddress);
}
private int getMaxIdMSSQL(string tableName, string colName)
{
SqlConnection con = new SqlConnection();
con.ConnectionString =
"server=localhost;database=MehrozeDSM;uid=sa;pwd=santana";
SqlDataReader dr;
SqlCommand cmd = new SqlCommand();
cmd.Connection = con;
cmd.CommandText = "Select max(" + colName + ")+1 from " + tableName;
con.Open();
dr = cmd.ExecuteReader();
dr.Read();
int maxId=(int)dr[0];
dr.Close();
con.Close();
return (maxId);
}
}
}
```