



Project Registration Number: WZ1096

COMPLETE DOCUMENTATION FOR THE HOME ENERGY MONITOR INTERFACE

Table of Contents

TOPIC	page
Project Description	
• General Description of Project.	1 - 2
Hardware / Software Details	
• Real-Time AC Voltage measurements.	3 - 6
• Real-Time AC Current measurements.	7 - 8
• WIZnet WIZ550io Ethernet Module.	9 - 10
• Exosite portal web site.	11 - 20
• Arduino Uno Board.	21 - 22
Operational Details	
• Overview of project operation.	23 - 25
• Photos of the completed assembly.	26 - 27
Project Hardware Information	
• Hardware Parts List	28
• Completed Project Schematic	29

Project Description: (General Description of Project)

This project measures real-time AC voltage and AC currents of an entire home. The data values calculated utilizing these real-time measurements are kilowatts (KW) and kilowatt-hours (KWH) in addition to the cost of using electricity. The measurements and calculated data values are sent to the Exosite Portal web site for viewing and data logging.

The components used in this project are installed into an active main AC voltage load center, as a result care must be taken when installing these components. The block diagram below illustrates the connections between the project's main components.

A 9vdc power supply provides power to the Arduino Uno board which converts this into 5vdc and 3.3vdc power. The 3.3vdc is used to power the WIZnet WIZ550io Ethernet Module and the 5vdc provides power for the status LEDs. An AC Voltage Transformer (VT) and AC Current Transformers (CT) are connected to the analog inputs of the Arduino Uno board to allow measurement of single phase voltage and AC currents. The WIZnet WIZ550io Ethernet Module communicates with the Arduino Uno board using the SPI communication protocol.

The data is sent to the Exosite Portal web site using the WIZ550io Ethernet Module as a gateway to the internet. An ethernet cable is run to the main ac voltage load center for this purpose. With this connection, the home energy usage can be accessed anywhere a PC can connect to the internet.

Block Diagram:

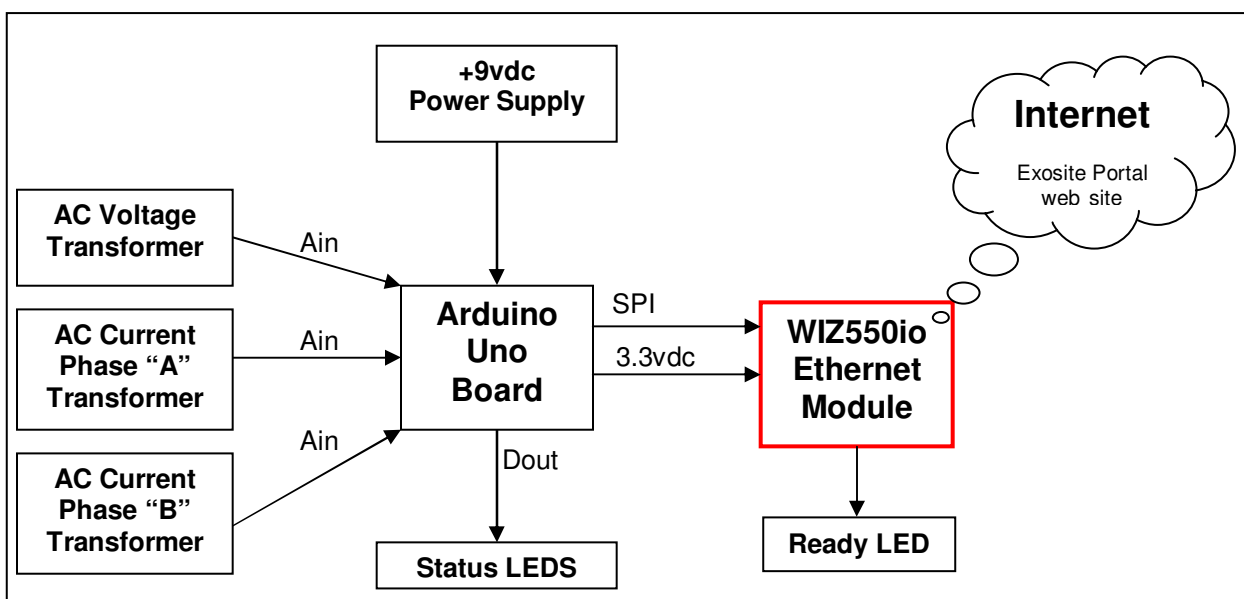


Photo 1 depicts the completed board. All components are wired together using a breadboard and jumper wires. The board is placed inside the main ac voltage load center. The following connections are required to make the board functional: Phase “A” and “B” CTs, an AC VT, 9vdc power supply and an ethernet cable. LEDs connected to the Arduino UNO board will turn “on” indicating a good connection/status of the transformers. The LED connected to the WIZ550io ethernet module will turn “on” when the module is ready and operational.

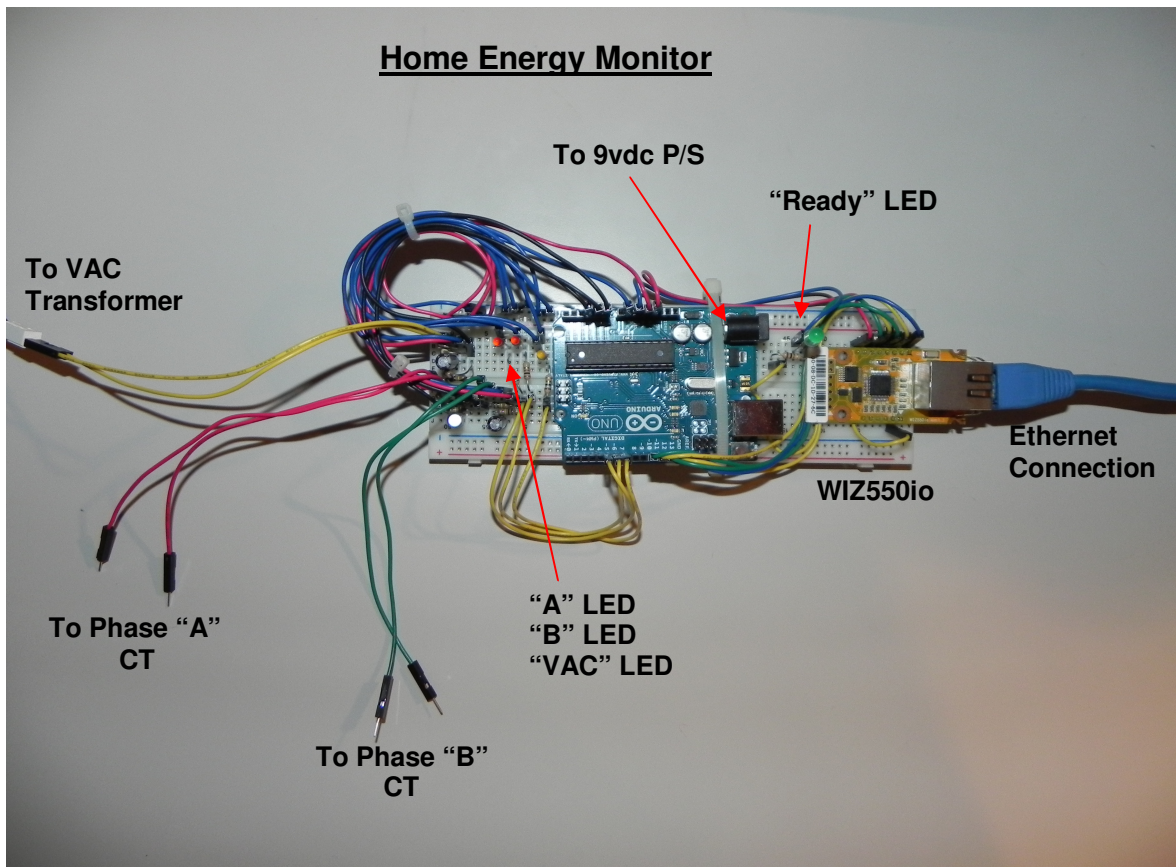
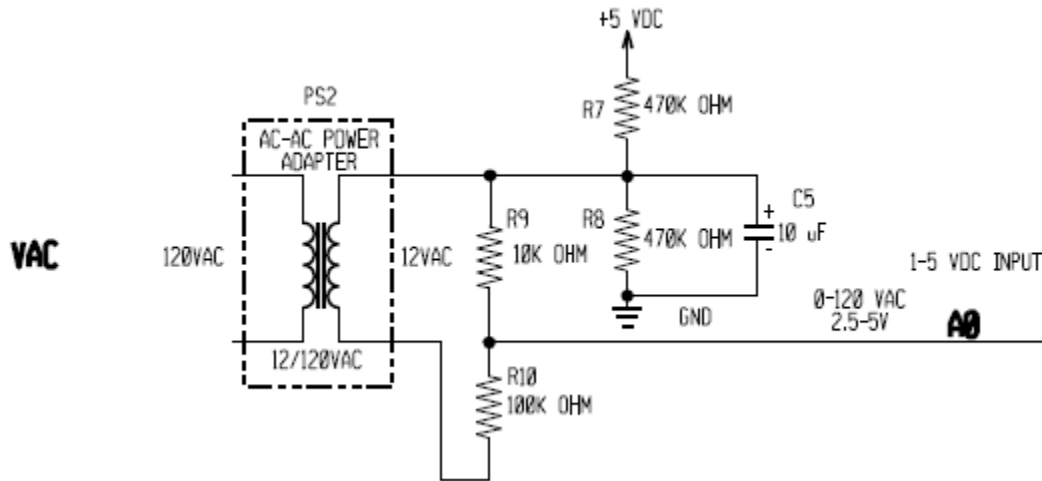


Photo 1:

Completed board showing component layout and wiring connections that need to be made.

Hardware / Software Details (Real-Time AC Voltage measurements)

The ac voltage reading is measured using a VT which has a 12vac output that is read by the Arduino board analog input. The VT is plugged into an AC outlet which measures the AC voltage of only 1 phase. See [Circuit Diagram 1](#) below for wiring details.



Circuit Diagram 1:

Hardware wiring connections of the AC Voltage Transformer to the Arduino Analog input.

The AC voltage waveform is measured by taking precise timed samples of the entire waveform. The highest reading is then determined to produce an accurate AC voltage reading. But before the waveform can be accurately measured, the AC zero crossing must be detected to ensure the full waveform is being properly detected and measured. See [Code Listing 1](#) for the zero crossing routine and [Code Listing 2](#) for the voltage measuring software routine.

Code Listing 1:

Routine to measure the AC waveform zero crossing.

```
//=====
void VAC_Zero_Cross()
//=====
{
    st=false;                // Indicator to exit while loop
    if ((VAC_present) & (loop1))
    {
        //Waits for the waveform to be close to 'zero' (500 adc)
        while(st == false)
        {
            delayMicroseconds(100);
            VACreading0 = analogRead(sensorPinA0);
            //check its range.
            if (VACreading0 = 511)
        }
    }
}
```

```

        {
            st=true;
        }
    }
}
loop1=true;
// VAC Present indicator LED
if (VAC_present)
{
    digitalWrite(VAC_ledPin, 1);
}
else
{
    digitalWrite(VAC_ledPin, 0);
}
}

```

Code Listing 2:

Routine to measure the AC waveform to determine an accurate AC voltage reading.

```

//=====
void Measure_VAC()
//=====
// Measure AC Voltage. (0-120vac = 0-5v output to ADC)
// 100 measurements every 20 milliseconds.(0.02/100=200usec/sample)
//-----
{
    VAC_present = false;
    for (LP1=0; LP1<100; LP1++)
    {
        delayMicroseconds(200);
        VACreading = analogRead(sensorPinA0); // Read AC voltage.
        if (VACreadingHI < VACreading) VACreadingHI = VACreading;
    }
//-----
// Measure AC Voltage.
//-----
    if (VACreadingHI < 505) VACreadingHI = 505;
    RT_VAC_Reading = VACreadingHI;
    if (RT_VAC_Reading > 700)
    {
        VAC_present = true;
    }
    if (VAC_present == false)
    {
        RT_VAC_Reading = 511; // NO VAC
    }
    VACreadingHI = 0;
}

```

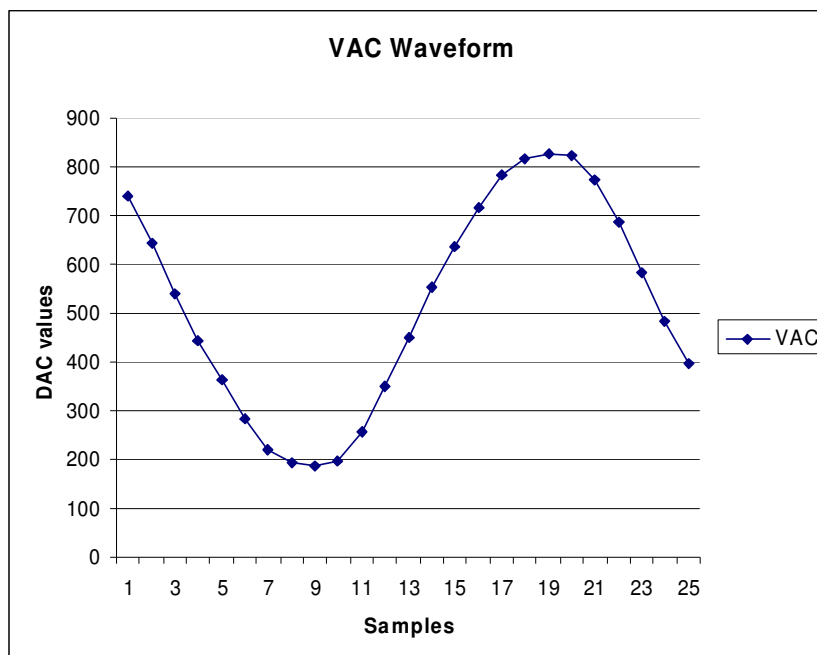
To determine if the analog input was in fact measuring the waveform accurately, another routine was used to send the readings to a PC serially and then using Microsoft excel to plot the values. See [Code Listing 3](#) below for the waveform measurement.

Code Listing 3:

Routine to measure the AC waveform by taking 25 samples and sending them serially to a PC to be plotted.

```
//=====
void Waveform_VAC()
//=====
{ // Get 25 AC voltage samples.
    if (VAC_present)
    {
        for (LP1=0; LP1<25; LP1++)          // Loop 25 times
        {
            delayMicroseconds(700);          // delay between
sampling.
            VACreading = analogRead(sensorPinA0); // Read AC voltage.
            VA_READING[LP1]=VACreading;          // LP1 -> 0-24.
        }
    }
//=====
// Send AC Voltage values to trend on PC via serial port.
//-----
    for (LP1=0; LP1<25; LP1++)          // Loop 25 times.
    {
        Serial.println(VA_READING[LP1]); // Send data to PC.
    }
}
}
```

The resulting plot ([plot 1](#)) shows that the waveform is being measured correctly and accurately. You can see from the data points collected that a uniform measurement is being taken throughout the waveform. As for the AC voltage measurement routine, 100 measurements are taken to accurately determine the AC voltage. The AC voltage waveform routine is only taking 25 measurements to get a sense for how accurately the waveform is actually being measured. Using a DC bias (2.5v level shifter) for the analog input, the sine wave will oscillate around 2.5vdc and remain positive. This will enable measurement of the entire AC voltage waveform as well as the AC current waveform. The 10 bit analog input at full range has 0-1024 ticks. With the DC bias (voltage divider) incorporated will only use 512 ticks. Consequently, the entire waveform is represented from 512 to 1024 ticks of the ADC. Therefore, the full range for the 120VAC measurement will represent 512 ticks of the ADC.

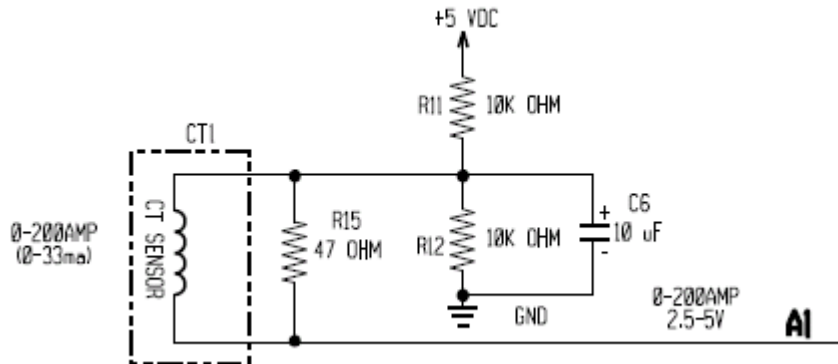


Plot 1:

This shows that the analog input is measuring the AC waveform correctly and accurately. A special feature of this project is that it will take 100 measurements per sampling of the AC voltage and AC currents to obtain the accurate readings. The waveform above is made with sampling only 25 AC voltage readings. This was used during the debug phase of the project.

Hardware / Software Details (Real-Time AC Current measurements)

The AC current of two phases are measured. The CTs used are rated at 0-200 amperes and produce 33ma per ampere. These signals are wired to the Arduino board analog inputs. See [Circuit Diagram 2](#) below for wiring details.



Circuit Diagram 2:

Hardware wiring connections of the AC Current Transformer to the Arduino Analog input.

The current waveform is measured in the same manner as the voltage waveform. As with the voltage measurement, the full range for the 200 AC current measurement will represent 512 ticks of the ADC. See [Code Listing 4](#) for the current measuring software routine.

Code Listing 4:

Routine to measure the AC waveform by taking 25 samples and sending them serially to a PC to be plotted.

```
//=====
void Measure_AMPS_A()
//=====
// Measure AC Current. (0-200amps = 0-5v output to ADC)
// 100 measurements every 20 milliseconds.(0.02/100=200usec/sample)
//-----
{
  if (VAC_present)
  {
    for (LP1=0; LP1<100; LP1++)
    {
      delayMicroseconds(200);
      AMPreading = analogRead(sensorPinA1); // Read AC phase "A" amps.

      if (AMPreadingHI < AMPreading) AMPreadingHI = AMPreading;
    }

    if (AMPreadingHI < 550)
    {
      AMPreadingHI=511;
      digitalWrite(A_AMP_ledPin, 0);
    }
    else
    {

```



```

        digitalWrite(A_AMP_ledPin, 1);
    }

    RT_A_AMP_Reading = AMPreadingHI;
    AMPreadingHI=0;
}
}

```

The CTs are installed by clipping them around the cables entering the main AC load center from the power company. See Photo 2 below for CT connections. The CTs used here have large openings to allow clipping around the incoming cables. If other CTs are used be sure to measure the diameter of these cables.

WARNING

Be Very Careful while working inside the energized main AC load center.

**Electrical shock can occur and could be deadly.
Proper electrical personal protective equipment must be worn.**



Phase "A" CT

Phase "B" CT

Photo 2:

Current Transformers connected around incoming 120VAC power cables inside the load center.

Hardware / Software Details (WIZnet WIZ550io Ethernet Module)

This project uses the WIZ550io Ethernet Module as a gateway for sending energy usage information to the internet. The WIZ550io will automatically obtain an IP address which makes it straightforward to set-up and use. See [Code Listing 5](#) for displaying the network information from this module using your PC serial port.

Code Listing 5:

Routine to read back WIZ550io network information.

```
//=====
// Initialize Ethernet communication.
//=====
Ethernet.begin();
delay(2000);
// read IP Address on WIZ550io.           (ex. 192.168.15.104)
// read SUBNET Address on WIZ550io.       (ex. 255.255.255.0)
// read DNS Server IP Address on WIZ550io. (ex. 192.168.15.1)
// read GATEWAY IP Address on WIZ550io.   (ex. 192.168.15.1)
Serial.println(Ethernet.localIP());
Serial.println(Ethernet.subnetMask());
Serial.println(Ethernet.dnsServerIP());
Serial.println(Ethernet.gatewayIP());
```

The MAC address of the WIZ550io module will need entered which will be unique to each project. The address can be found on the module board and is entered in the macData variable below:

```
byte macData[] = {0x00, 0x08, 0xDC, 0x1D, 0x27, 0x6C};
```

This allows the module to connect to the internet and communicate with the web site.

Communication between devices use the SPI protocol which is quick enough to transmit and receive data to and from the web site. Make certain that the MISO pins from these devices are connected together as well as the MISI pins. If these are reversed which makes sense (out -> in and in <- out), the module will not accept commands and will not be obvious what is occurring or not occurring.

MISI → MISI

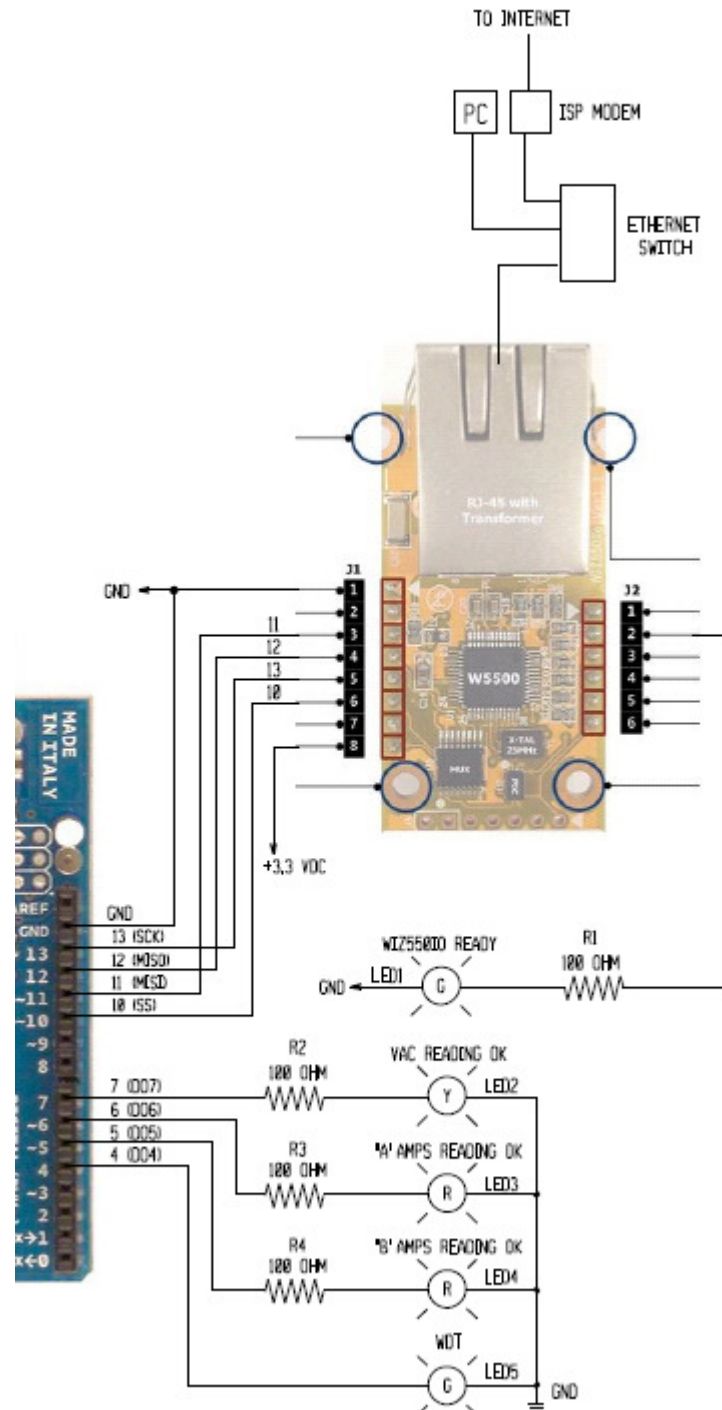
MISO → MISO

The module receives its power from the Arduino Uno board 3.3vdc output. After the module successfully completes the on-board self test, the Ready LED will turn “on” signifying it is ready to accept commands. See [Circuit Diagram 3](#) for wiring connection details.

Circuit Diagram 3:

Wire connections for the WIZnet WIZ550io Ethernet Module.

- Connect the ethernet cable from your ethernet switch and the module will be ready to use!



Hardware / Software Details (Exosite portal web site)

The Exosite routine for sending and receiving data from the web site can be seen in [Code Listing 6](#) below. The cikData string from the Exosite web site will need entered which is also unique to each project. The actual readings and calculated value variables will need set-up at both ends for communication with the web site to be successful.

Code Listing 6:

Routine to write/read data to/from Exosite portal web site.

```
*=====*
* Configuration Variables
*=====*
String cikData = "8cd2f79937f2b89bdcc2c6003304103db339704d"; //exosite
cikData
byte macData[] = {0x00, 0x08, 0xDC, 0x1D, 0x27, 0x6C};
//WIZ550io MAC.
// Use these variables to customize what data sources are read and
written to.
String readString = "CostperKWH";
String writeString = "rt-vac=";
String writeString1 = "rt-a_amp=";
String writeString2 = "rt-b_amp=";
String writeString3 = "rt-total_amps=";
String writeString4 = "rt-kw=";
String writeString5 = "rt-kwh=";
String writeString6 = "rt-kwh_cost=";
String returnString;
String rstr;

//-----
// Send values to the exosite portal every 60 seconds.
//-----
void Connect_with_Exosite()
{
    upt = upt + 1;
    // Perform 1 min Calcs
    Total_Amps = A_amp + B_amp;
    RT_KW = (VAC * Total_Amps) / 1000.0;
    RT_KWH = RT_KWH_prev + (RT_KW / 60.0); // update every 60 seconds.
    RT_KWH_prev = RT_KWH;
    // convert to long value for web site.
    VAC_L = VAC * 10;
    A_amp_L = A_amp * 10;
    B_amp_L = B_amp * 10;
    Total_Amps_L = Total_Amps * 10;
    RT_KW_L = RT_KW * 1000;
    RT_KWH_L = RT_KWH * 100;

    // Send Data to Web Site, update every minute.
    if (exosite.writeRead(writeString+String(VAC_L), readString,
returnString)){}
```

```

    if (exosite.writeRead(writeString1+String(A_amp_L), readString,
returnString)){}
    if (exosite.writeRead(writeString2+String(B_amp_L),
readString,returnString)){}
    if (exosite.writeRead(writeString3+String(Total_Amps_L), readString,
returnString)){}
    if (exosite.writeRead(writeString4+String(RT_KW_L), readString,
returnString)){}

    // update hourly
    if (upt >= Update_Int) // update web site every 60 minutes.
    {
        upt=0;
        if (exosite.writeRead(writeString5+String(RT_KWH_L), readString,
returnString)){}
        // convert string to float.
        rstr = returnString.substring(11);
        Cost_per_KWH_L = rstr.toInt();
        Cost_per_KWH = (float)Cost_per_KWH_L / 10000.0;
        RT_KWH_Cost = RT_KWH * Cost_per_KWH;
        // calculate the Cost per KWH.
        RT_KWH_Cost_L = RT_KWH_Cost * 10000;
        // Send "Cost per KWH" to Web site.
        if (exosite.writeRead(writeString6+String(RT_KWH_Cost_L), readString,
returnString)){}
        RT_KWH = 0;
        RT_KWH_L = 0;
        RT_KWH_prev = 0;
        RT_KWH_Cost = 0;
        RT_KWH_Cost_L = 0;
    }
}

```

See Photo 3 below to begin the registration/set-up process for a free developmental account on the Exosite Portal web site. Use the link below to go to the sign up page:

<https://support.exosite.com/registration>

Photo 3:

Exosite Portal Web Site – Sign Up for a new account.

Sign up to Exosite Support

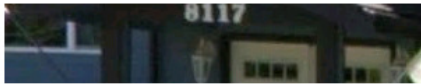
Please fill out this form, and we'll send you a welcome email to verify your email address and log you in.

Your full name *

Your email *

Your Twitter

Please verify text *



[Two other words please](#) [I want audio instead](#)

[Cancel](#)

NOTE: To view the “Home Energy Monitor” Exosite Portal Web Site, follow the link below: <https://portals.exosite.com/views/3093978846/1232274221>

Data to be displayed must be entered on the Exosite Portal web site data page and match corresponding variables in the Arduino code. See the example below for setting up the real-time ac voltage reading to be displayed on the web site. See [Photo 4](#) (p.16) for all data variables used in this project.

Example 1:

Display data on the Exosite Portal Web Site.

(1) Enter data variables in the Arduino code and (2) on the web site data page, (3) Create a widget to display the data.

1. Arduino Code:

```
String writeString = "rt-vac=";  
  
// convert to long value for web site.  
VAC_L = VAC * 10;  
  
// Send Data to Web Site, update every minute.  
if (exosite.writeRead(writeString+String(VAC_L),readString,  
returnString)) {}
```

2. Exosite Portal Web Site Data Entry:

The screenshot shows the 'Data Information' page in the Exosite Portal. It is divided into sections: 'Data Update', 'Source Info', and 'Retention'. The 'Data Update' section contains fields for Name, Current, Value, Units, Format, and Storage. The 'Source Info' section contains fields for Device, Alias, RID, and Calculation. The 'Retention' section contains radio buttons for 'infinity' and 'custom', and fields for Duration and Count. Arrows point from a list of instructions to specific fields: 'rt-vac' in the Name field, 'VAC' in the Units field, 'rt-vac' in the Alias field, 'Divide' and '10' in the Calculation dropdowns, and '1' in the Count field.

Data Information	
Data Update	
Name:	rt-vac
Current	120.3
Value:	
Units:	VAC
Format:	string
Storage:	21.93 KB
Source Info	
Device:	HEM
Alias:	rt-vac
RID:	2538cd3e226183cb3198464cf1cab0d229b48bfa
Calculation:	Divide 10
	NA
Retention	
	infinity custom
Duration:	<input checked="" type="radio"/> <input type="radio"/> hours
Count:	<input type="radio"/> <input checked="" type="radio"/> 1 data points




Enter:

1. Variable Name
2. Units
3. Format = string
4. Alias = rt-vac
5. Calculation: /10
6. Count = 1 data pt

3. Exosite Portal Web Site Widget Entry: (Gauge)

Click on the “Add Widget” button located at the top right of the web site. Select the Gauge widget type. Enter the block title then click on the “Continue” button. After the information is entered below click on the “Save” button.

Widget Set-Up Page:

Real Time Meter - VAC   

Block Title:

Real Time Meter - V.

Widget Type:

Gauge

Min Level:

0


Max Level:

240


Low Level:

0

Color:

 #41C4DC


Mid Level Color:

 #FFFFFF

High Level:

250

Color:

 #A91E27

Default Active Data:

rt-vac

Set Caller:

-NONE-

Refresh Rate:

10

Seconds

(0 seconds = no refresh)

CANCEL

SAVE

Resulting Gauge Widget:

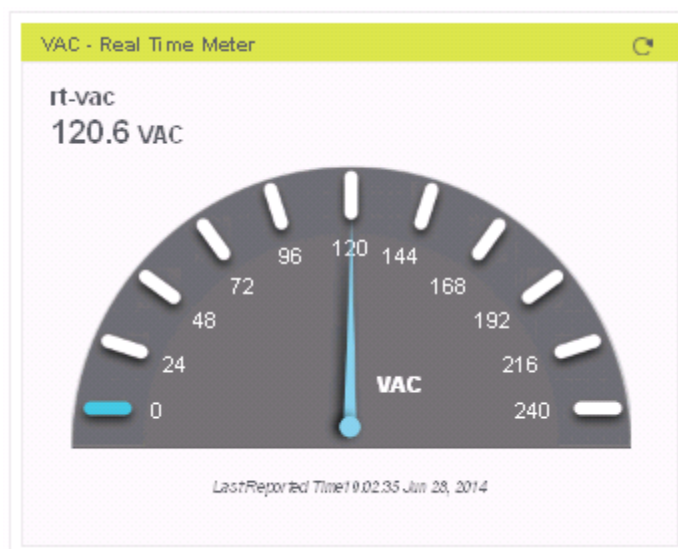


Photo 4:**Exosite Portal Web Site Data Page – Set-Up Data Variables.**

Data + Add Data				
Name ▲	Alias	Last Value	Unit	Last Reported Time
Portal: Jim Abraham				
Device: <u>HEM</u>				
CostperKWH	CostperKWH	585	\$	19:09:20 Jun 14, 14 America/New_York
rt-a_amp	rt-a_amp	0.0	AC AMPS	17:47:09 Jun 29, 14 America/New_York
rt-b_amp	rt-b_amp	12.5	AC AMPS	17:47:10 Jun 29, 14 America/New_York
rt-kw	rt-kw	1.504	KW	17:47:11 Jun 29, 14 America/New_York
rt-kwh	rt-kwh	3.28	KWH	17:14:55 Jun 29, 14 America/New_York
rt-kwh_cost	rt-kwh_cost	0.1922	\$	17:14:55 Jun 29, 14 America/New_York
rt-total_amps	rt-total_amps	12.5	AC AMPS	17:47:11 Jun 29, 14 America/New_York
rt-vac	rt-vac	120.3	VAC	17:47:09 Jun 29, 14 America/New_York

From the real-time AC voltage and AC current measurements, KWs and KWHs and the cost of using electricity can be calculated. The real-time AC voltage and AC currents are measured every two seconds. These are used to construct one minute average readings for calculating a hourly KWH as well as the cost associated with the electric usage. The real-time readings and calculations are sent to the Exosite Portal web site for viewing.

From the Exosite Portal web site:

- View one minute update of real-time readings (AC voltage, AC current, Kilowatts).
- View hourly and daily KWHs and costs.
- Enter the cost per KWH from your electric bill to check if the power company is accurately measuring your POWER USAGE!

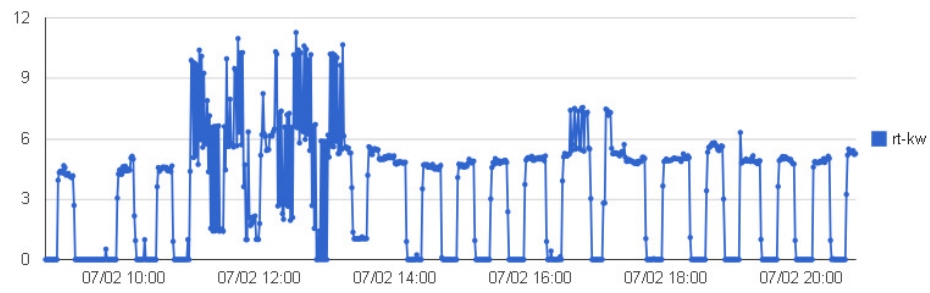
For real-time AC measurements see Photo 5 for gauge widgets and Photo 6 for KWH information.

Photo 5:
Exosite Portal Web Site - Real-Time Meters. (VAC, AC AMPS, KWs)

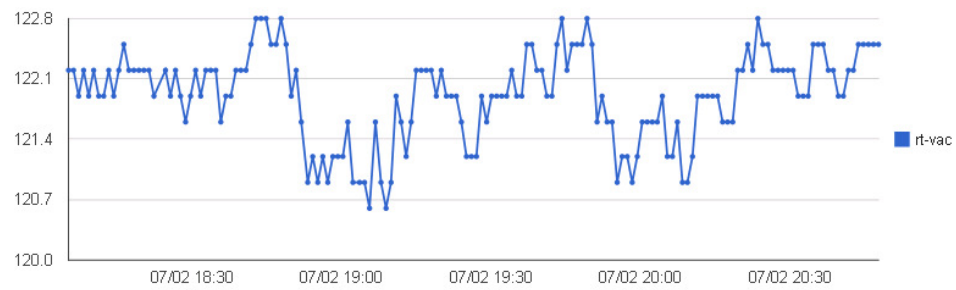


See next page for Real-Time AC Measurement Charts.

KILO-WATTS



AC VOLTAGE



PHASE "A and "B" AMPS

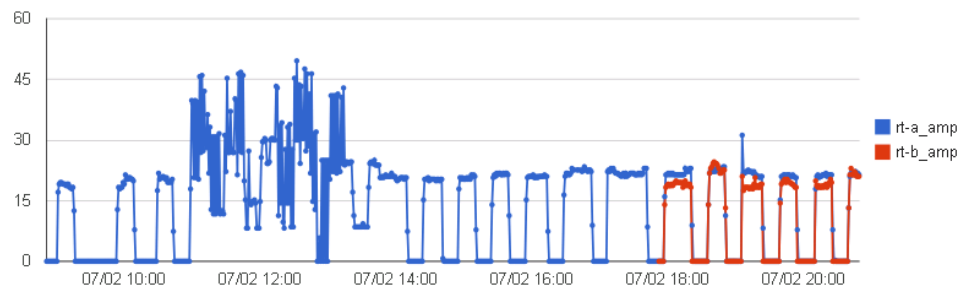
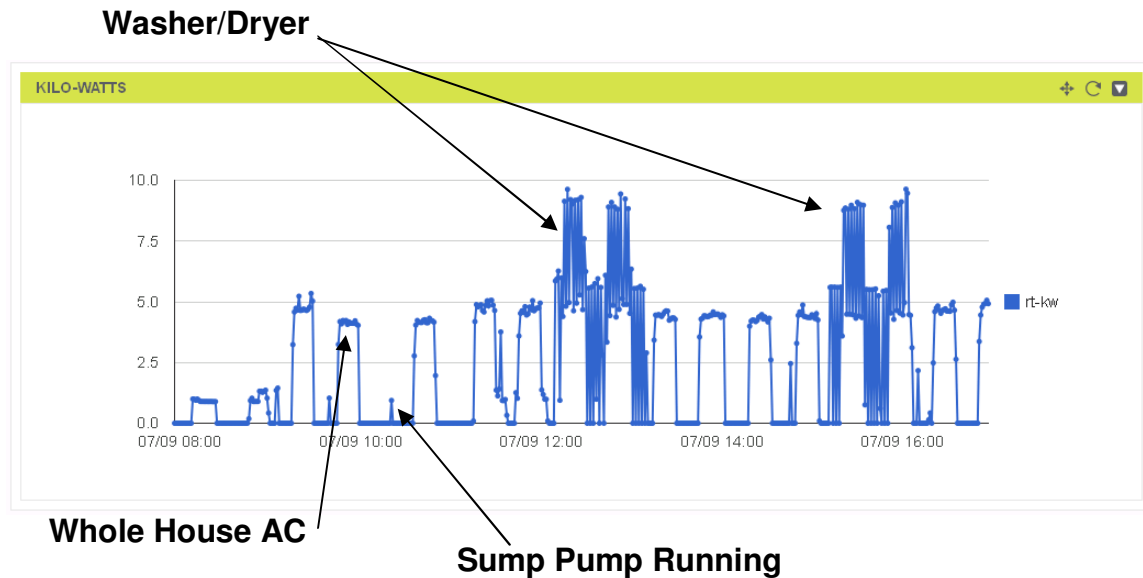


Photo 6:
Exosite Portal Web Site - KWH information



It's very interesting to visualize how appliances operate and can detect if one needs to be replaced or not working properly. This is a great project to learn more about your power usage in the home and what appliances or people in your household use the most electricity.



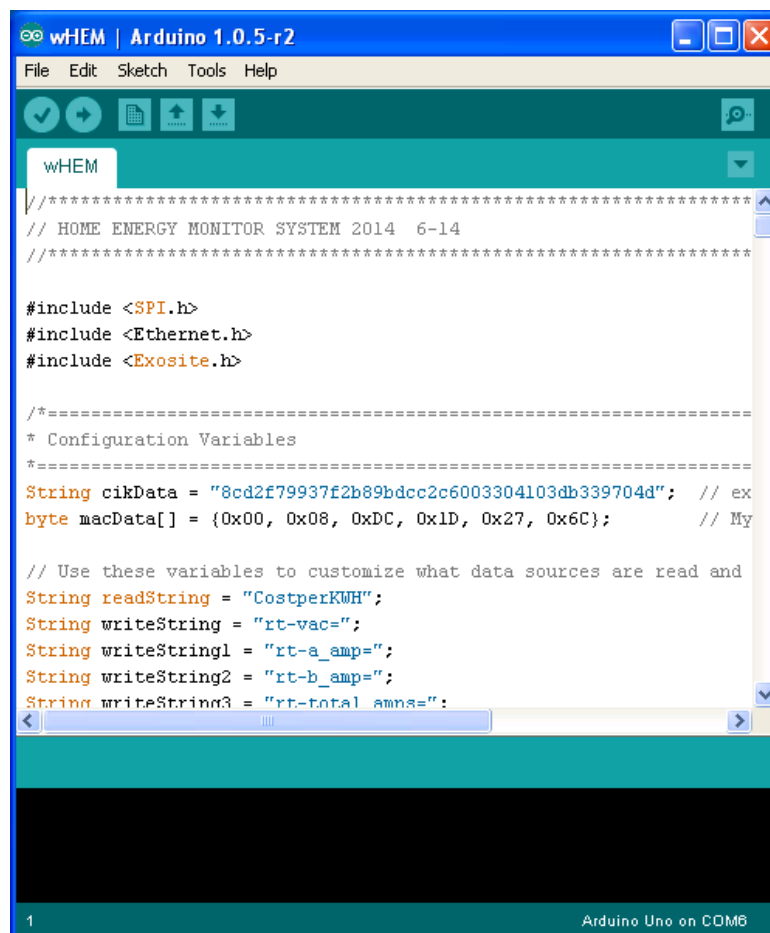
From the chart one can observe the whole house air conditioning cycling throughout the day (this is similar with heating but larger KWs are measured). When the washer/dryer is running it is apparent from the cycling of the KWs in a short period of time (this is similar with the dishwasher as well but with smaller KWs measured). Also the small spikes indicated that the sump pump was running. The water heater is unfortunately on a separate breaker and is not included in these readings. Although, would be valuable in tracking hot water usage.

Hardware / Software Details (Arduino Uno Board)

The Arduino Uno Board is used as the interface between collecting real-time power measurements and sending data to the web site. The board was programmed using the Arduino software. See [Photo 7](#) below for the Arduino software application. During software development, the USB port was connected to the board for easy software downloads. Troubleshooting was conducted by sending serial commands back through the same USB port and using the serial monitor program included with the application (located under tools drop down list). The application is free for download. The project file loaded for this project is named: **wHEM.ino**.

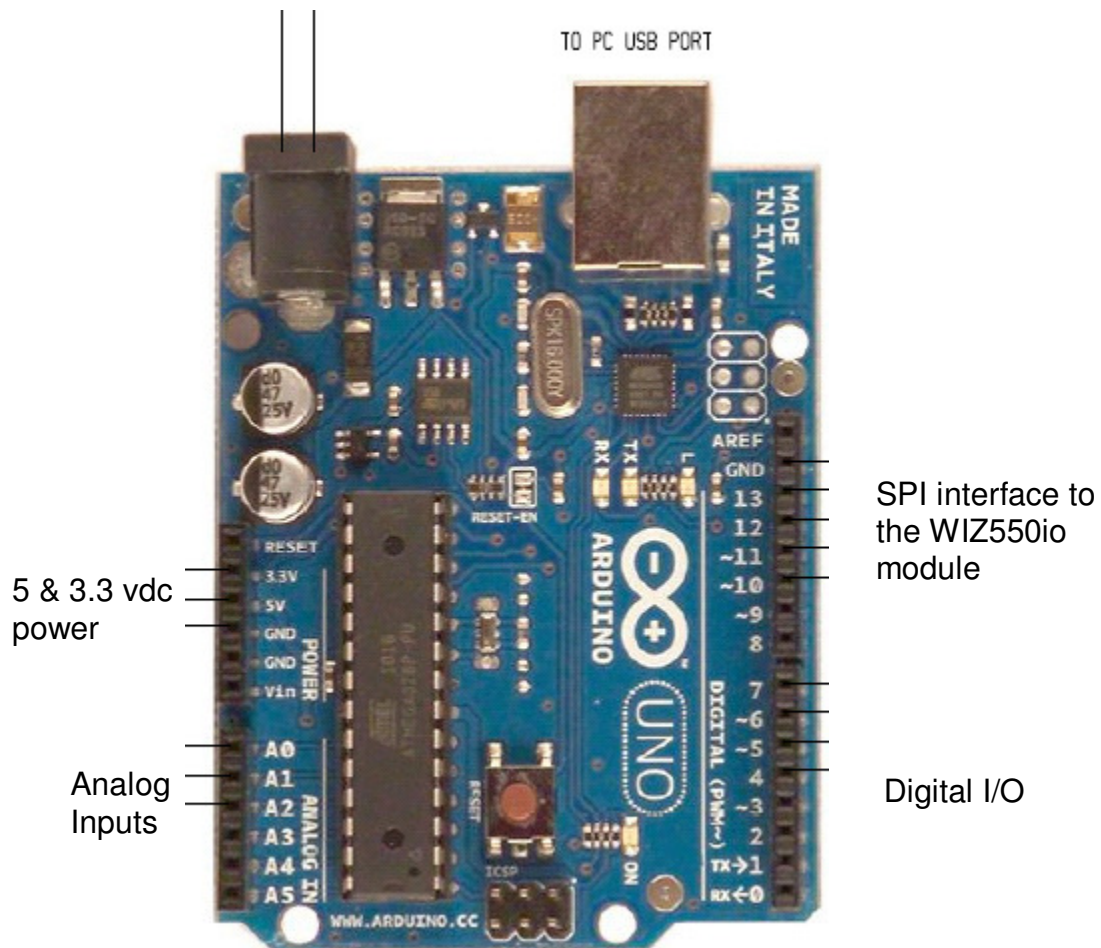
Photo 7:

Arduino software application is shown here with the wHEM.ino program loaded.



The Arduino Uno Board is shown below in Circuit Diagram 4. Top of the board is a USB port to connect to your PC. When connected it will also supply power to the board using the PC's 12vdc power supply. A power connection is also included to apply power to the board when development is complete. The board headers were used for making connections easily to the other system components.

Circuit Diagram 4:
Arduino Uno Board.



Operational Details (Overview of project operation)

The main loop program running on the Arduino Uno board is listed below ([Code Listing 7](#)). The AC voltage and currents are measured and accumulated for one minute. After which the one minute averaged readings are sent to the Exosite Portal Web Site.

AC Voltage Calculation:

$$VAC = (120.0 * (vacread2s - VACreading0)) / 374.0;$$

Where:

- vacread2s = ADC average two second reading.
- VACreading0 = ADC zero voltage crossing value (511).
- Using a 10-bit ADC: 120vac = 885 ticks, 0vac = 511 ticks.

AC Current Calculation:

$$A_amp = (100.0 * (A_ampread2s - VACreading0)) / 256.0;$$

Where:

- A_ampread2s = ADC average two second reading.
- VACreading0 = ADC zero voltage crossing value (511).
- Using a 10-bit ADC: 100amp = 767 ticks, 0amp = 511 ticks.

Code Listing 7:

Main Loop Routine to monitor AC voltage and current sensors.
- Every minute will send real-time data to the Exosite Portal Web Site.

```
/*=====*
* Main loop function.
*=====*/
/
void loop()
{
  VAC_Zero_Cross();
  Measure_VAC();          // Get VAC readings

  if (VAC_present)
  {
    VAC_Zero_Cross();
    Measure_AMPS_A();     // Get A Amp readings

    VAC_Zero_Cross();
    Measure_AMPS_B();     // Get B Amp readings

    // reading count
    vacreadcnt = vacreadcnt + 1;
    // accumulate readings
    vacread2s = vacread2s + RT_VAC_Reading;
    A_ampread2s = A_ampread2s + RT_A_AMP_Reading;
    B_ampread2s = B_ampread2s + RT_B_AMP_Reading;
    if (vacreadcnt > 1)
    {
      vacread2s = vacread2s / 2;
      A_ampread2s = A_ampread2s / 2;
      B_ampread2s = B_ampread2s / 2;
    }
  }
}
```



```

}
// make calculations and send data to web site.
// Every 60 seconds - Get 2 seconds readings -> avg and send to web site.
if (vacreadcnt >= Update_Int)
{
    VAC = (120.0 *(vacread2s - VACreading0)) / 374.0;
    vacreadcnt=0;
    vacread2s=0;

    // Get Phase A AMP readings
    A_amp = (100.0 *(A_ampread2s - VACreading0)) / 256.0;
    A_ampread2s=0;

    // Get Phase A AMP readings
    B_amp = (100.0 *(B_ampread2s - VACreading0)) / 256.0;
    B_ampread2s=0;

    // write/read data to/from web site (Every 60 seconds).
    Connect_with_Exosite();
}
digitalWrite(WDT_ledPin, 0); // watch dog timer led
delay(950);
digitalWrite(WDT_ledPin, 1);
}
} //loop

```

KW and KWH Calculations:

Watts are calculated by adding phase “A” and “B” amps together and then multiplying by the AC voltage. Dividing by 1000 will convert the Watts to KWs. This is performed every minute and sent to Exosite Portal Web Site.

Also, KWHs are calculated by accumulating the KWs calculated every minute and sending this information to the web site every hour. See [Code Listing 8](#) below. Note the Exosite Portal Web Site can only read integer values, so the data is converted to long integers. The values are multiplied to get the same accuracy.

Code Listing 8:

Code snippet of KW and KWH Calculations.

- Calculations are located in the “void Connect_with_Exosite()” routine.

```

Total_Amps = A_amp + B_amp;
RT_KW = (VAC * Total_Amps) / 1000.0;
RT_KWH = RT_KWH_prev + (RT_KW / 60.0); // update every 60 seconds.
RT_KWH_prev = RT_KWH;
RT_KWH_Cost = RT_KWH * Cost_per_KWH;

// convert to long value for web site.
VAC_L = VAC * 10;
A_amp_L = A_amp * 10;
B_amp_L = B_amp * 10;
Total_Amps_L = Total_Amps * 10;
RT_KW_L = RT_KW * 1000;
RT_KWH_L = RT_KWH * 100;
RT_KWH_Cost_L = RT_KWH_Cost * 10000;

```

The Completed Board Operation:

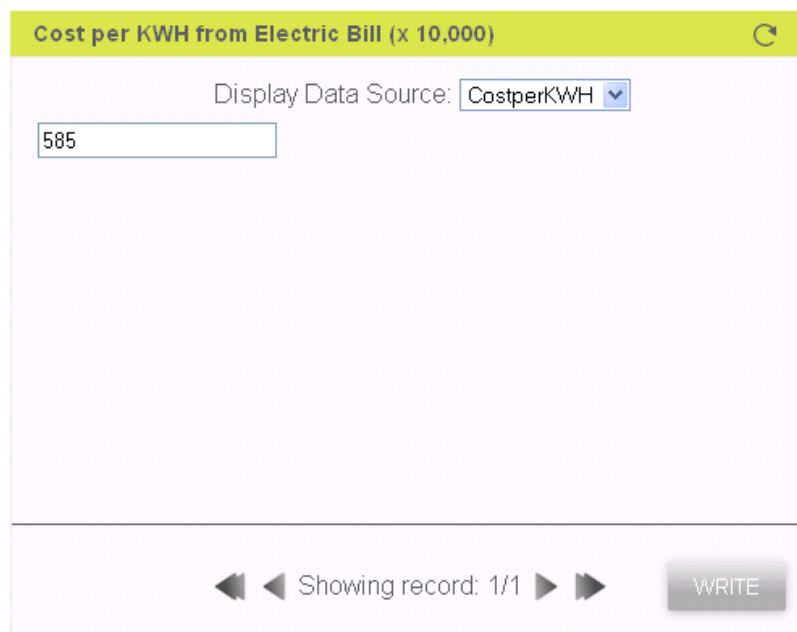
After the completed board is powered up and running, the WDT LED (watch dog timer led) will start to blink, indicating the program inside the Arduino board is running and the transformers and WIZ550io module is communicating with the Arduino board. The VAC LED will turn “on” indicating that the AC voltage is present and within range. The phase “A” and “B” LEDs will turn “on” when current is detected. When the AC voltage is present, AC measurements are taken every two seconds and one minute average readings are sent to the Exosite Portal Web Site.

The WIZnet WIZ550io Ethernet Module Operation:

The WIZ550io module is the gateway to the internet and to the Exosite Portal Web Site. This module will auto configure itself and will handle all internet communications. Sending data to the web site is performed with minimal commands and effort.

The Exosite Portal Web Site Operation:

To view the real-time data, log onto the web site dashboard and observe the measurements updating in the widgets and charts. Also, to change the cost per KWH, enter new costs from the web site.

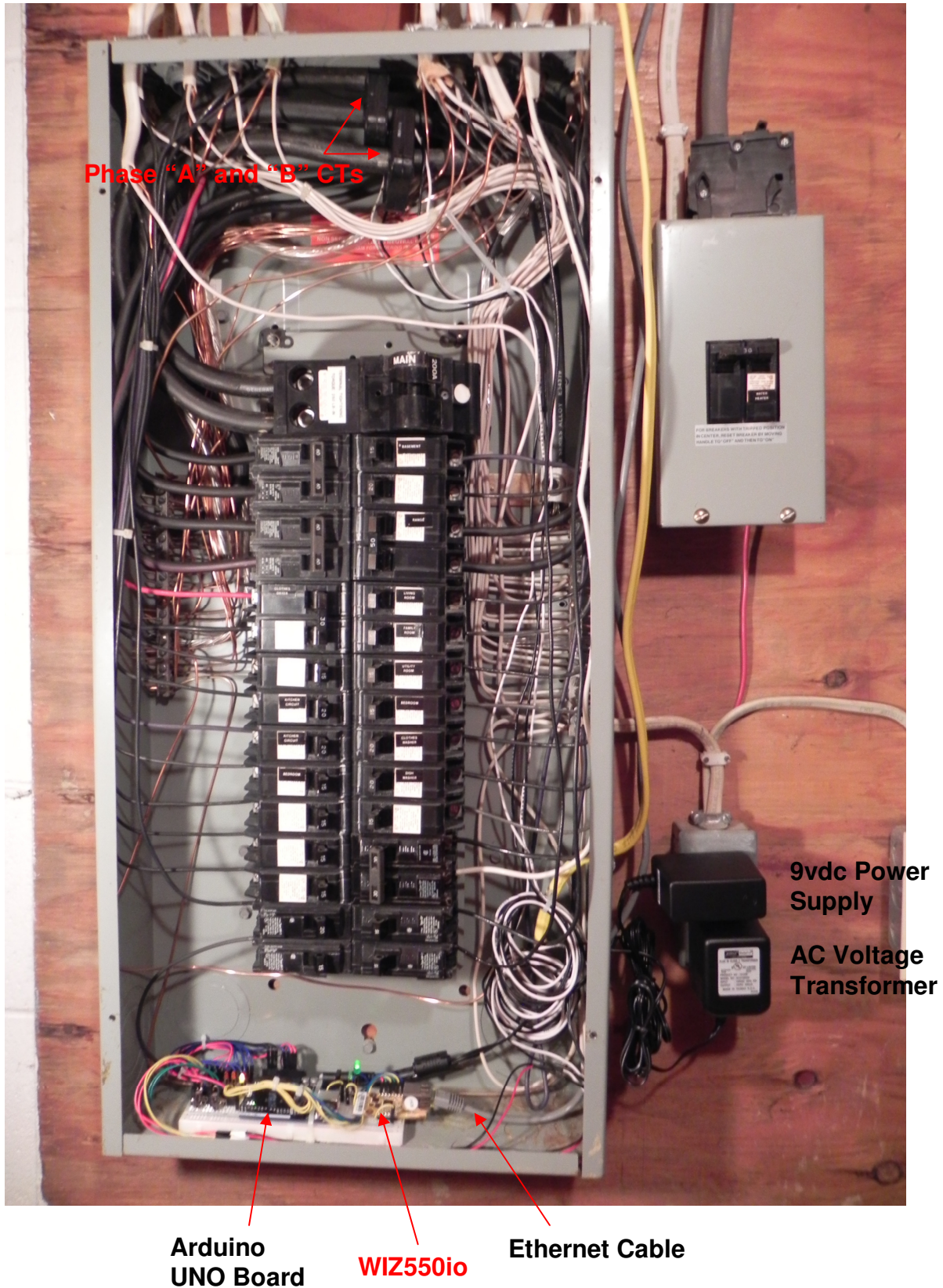


The screenshot displays a web interface for managing energy cost data. At the top, a yellow header bar contains the title "Cost per KWH from Electric Bill (x 10,000)" and a circular refresh icon. Below the header, the main content area has a light purple background. It features a label "Display Data Source:" followed by a dropdown menu currently set to "CostperKWH". A text input field below this contains the value "585". At the bottom of the interface, there is a navigation bar with left and right arrow icons, the text "Showing record: 1/1", and a "WRITE" button.

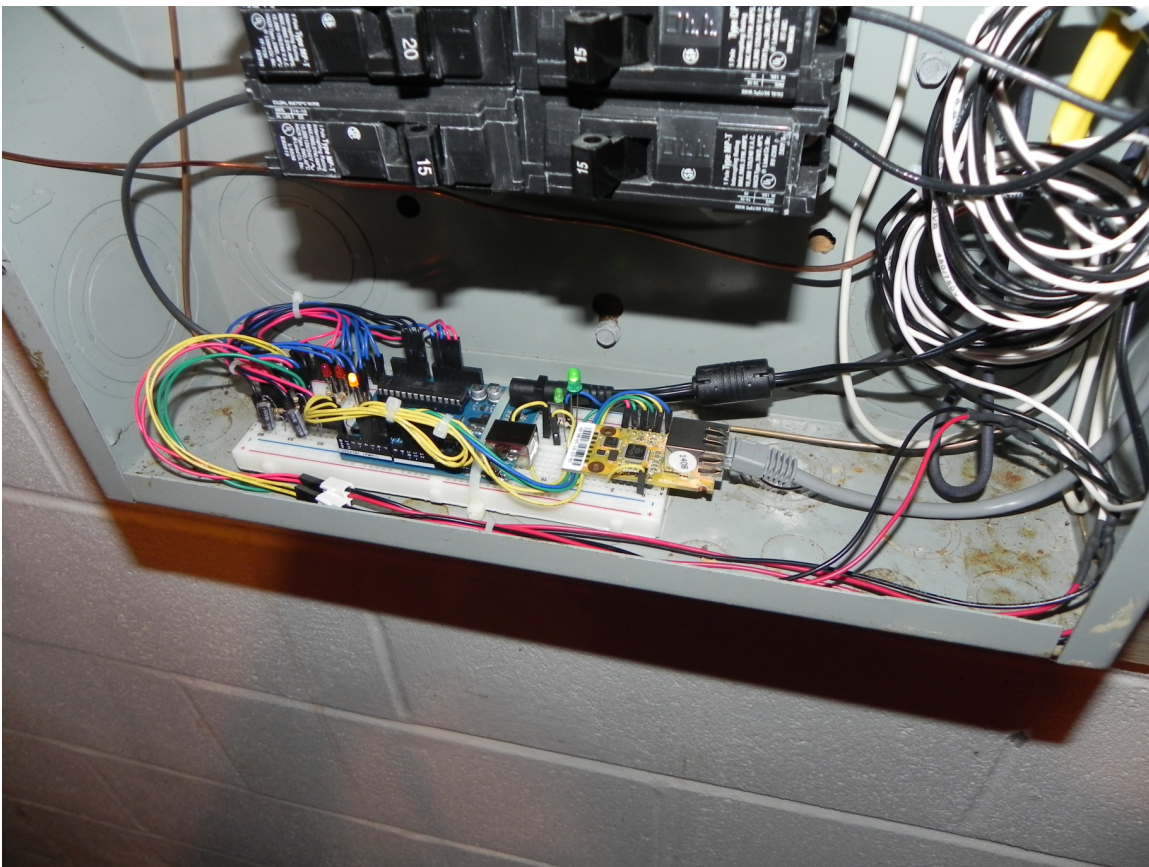
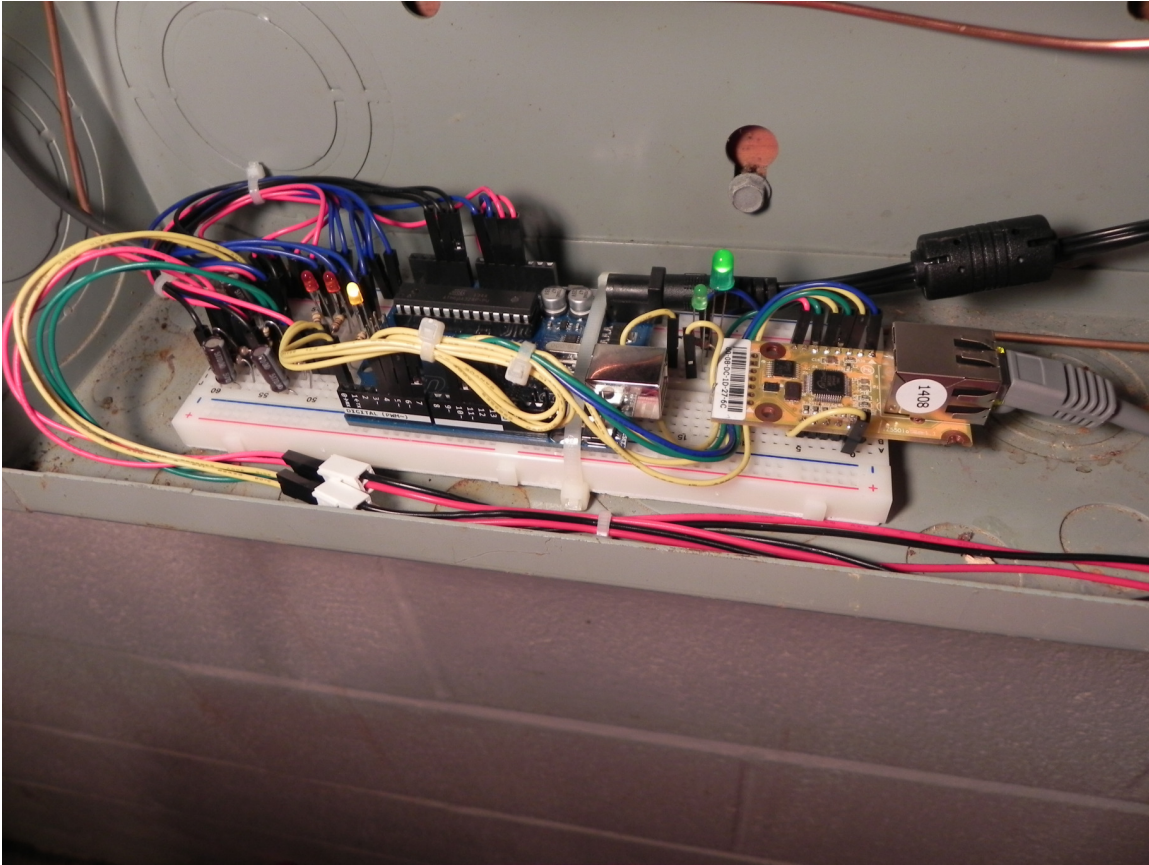
Operational Details (Photos of the Completed Assembly)

The completed project installed inside the Main AC Load Center:

- Note the cover can be safely re-attached to load center.



Completed Board inside Main AC Load Center:



Project Hardware Information (Hardware Parts List)

The major parts utilized in developing this project included a WIZnet WIZ550io Ethernet Module, Arduino Uno Board R3, +9vdc power supply, AC voltage transformer and AC current transformers (2 phases). Below is the complete parts list for this project.

HEM Hardware Parts List:

Qty	Component Name	Part Description	Part Numbers	Manufacturer
1	U1	WIZnet WIZ550io Ethernet Module	WIZ550io	WIZnet
1	U2	Arduino Uno Board R3	DEV-11021	Spark Fun
1	PS1	+9vdc Power Supply	2173132	Jameco
1	PS2	AC-AC Voltage Transformer	101258	Jameco
2	CT1, CT2	0-200 AMP AC Current Transformer	SCT-19-000	elecfreaks.com
2	LED1, 5	Green LED	697629	Jameco
2	LED3, 4	Red LED	2006713	Jameco
1	LED2	Yellow LED	697688	Jameco
4	R1-4	100 ohm resistor	690620	Jameco
5	R9, 11-14	10k ohm resistor	691104	Jameco
1	R10	100k ohm resistor	691340	Jameco
2	R8	470k ohm resistor	691500	Jameco
2	R15-16	47 ohm resistor	690540	Jameco
3	C5-7	10uF Capacitor	29891	Jameco
1		CAT 5 Ethernet cable. (15 ft)	201726	Jameco
3		2C cable connector (for transformers)	CON-242	All Electronics
1 pack		6" Jumper Wires	JMX6-100	All Electronics

Project Hardware Information (Completed Project Schematic)

