ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

ORGANISATION OF ISLAMIC COOPERATION (OIC)

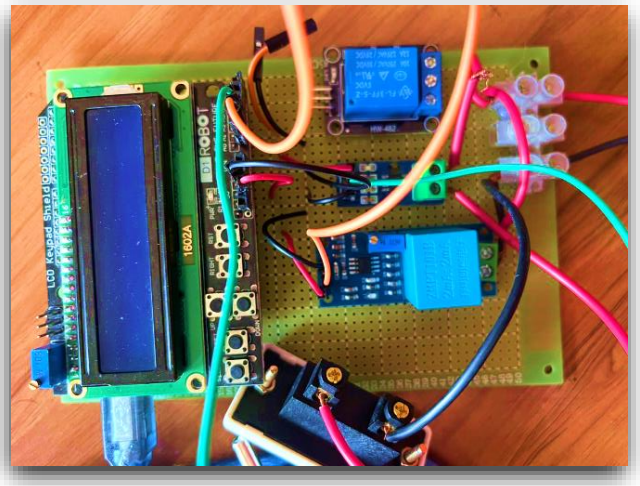
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Course No: EEE 4518

Course Title: Electrical and Electronic Workshop

Project Name:

Implementing a smart energy meter that includes cost calculation and over consumption protection.



Group Name: SadEEE

Name	**Id**
Fabbiha Bushra	200021105
Zerin Yeasmin	200021108
Nahadia Tabassum Oyshi	200021112
Md. Shariar Hossain Sadi	200021134
Adnan Shariar Abeer	200021140

Table of Contents

Table of figures:

Objective:

Implementing a smart energy meter that includes cost calculation,

and over consumption protection.

Description:

The purpose of this project was to receive input current from any electrical equipment and then feed it through the smart energy meter, which would calculate current and voltage to determine power use and anticipated electricity cost.

This meter is compatible with the electrical appliances we use every day, such the oven, refrigerator, TV, and others. The power measurement and how much electricity that specific gadget is using will be provided after that. Additionally, it will show us the per-unit cost estimate and the overall cost of that specific gadget during the time in question. With such information, we can estimate how long the device is going to operate, how much it will cost, and whether we should turn it off if it is using too much power.

Overall, this energy meter will serve as a unit of measurement for the majority of individuals who are not electrically literate but who can nevertheless compute the paid price and the time frame according to their own understanding.

Features:

- Measure power and real-time electrical unit usage and provide users or the general public with cost projections.
- Gives the projection of how much longer the device should function in terms of prepaid current and cost estimation.
- A device will have a current or price cap set for it, expressed, in KWh.
- Task navigating display. All the features will be shown and controlled in display.

Problem Space Overview:

Before jumping to the algorithm, we would like to give the idea about the problem space scenario so that it would be more convenient for the readers to understand ins and outs of the algorithm and upcoming description.

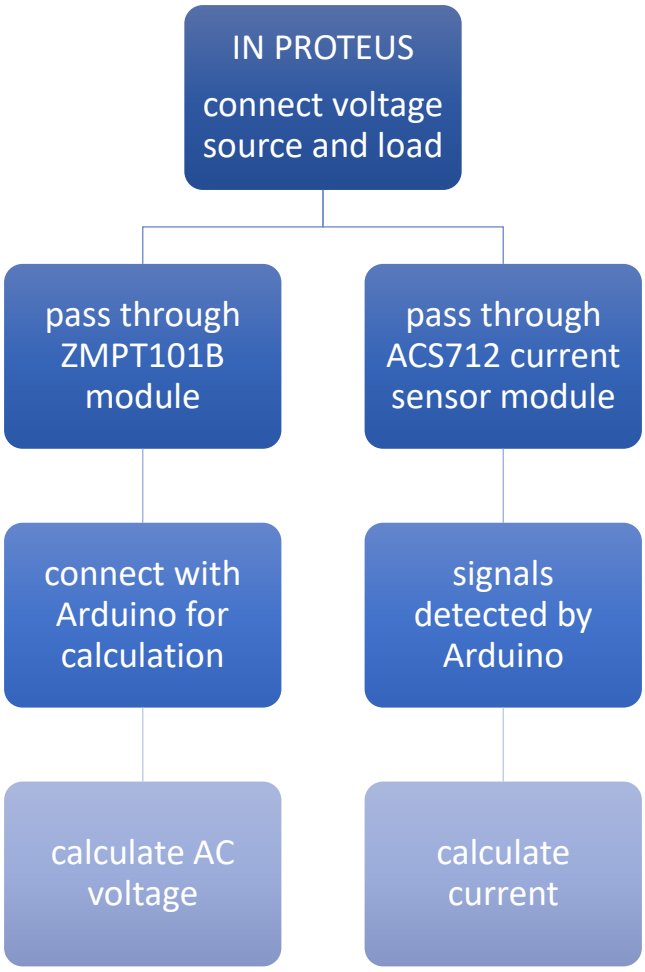
We can experiment with many equipment such as a heater, iron machine, microwave, refrigerator, AC, and so on, but for safety reasons, we will collect data using- i) table fan and

ii) light bulb.

iii)hair dryer.

We will simulate the project both in software and hardware.

Algorithm (Software):

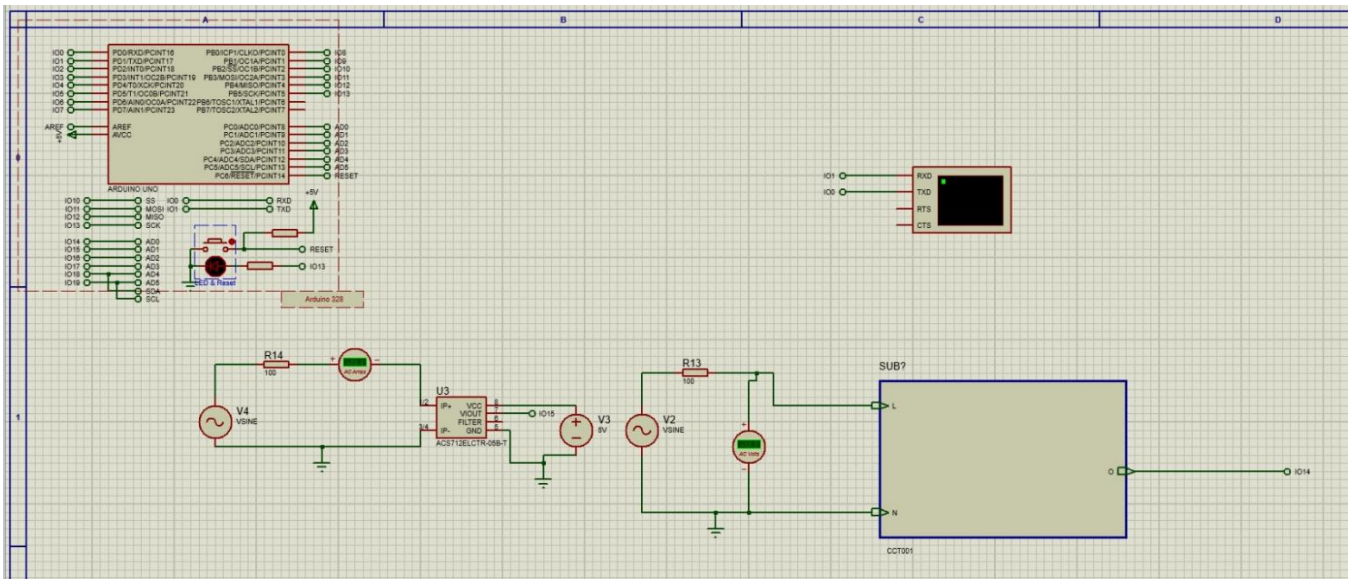


Software:

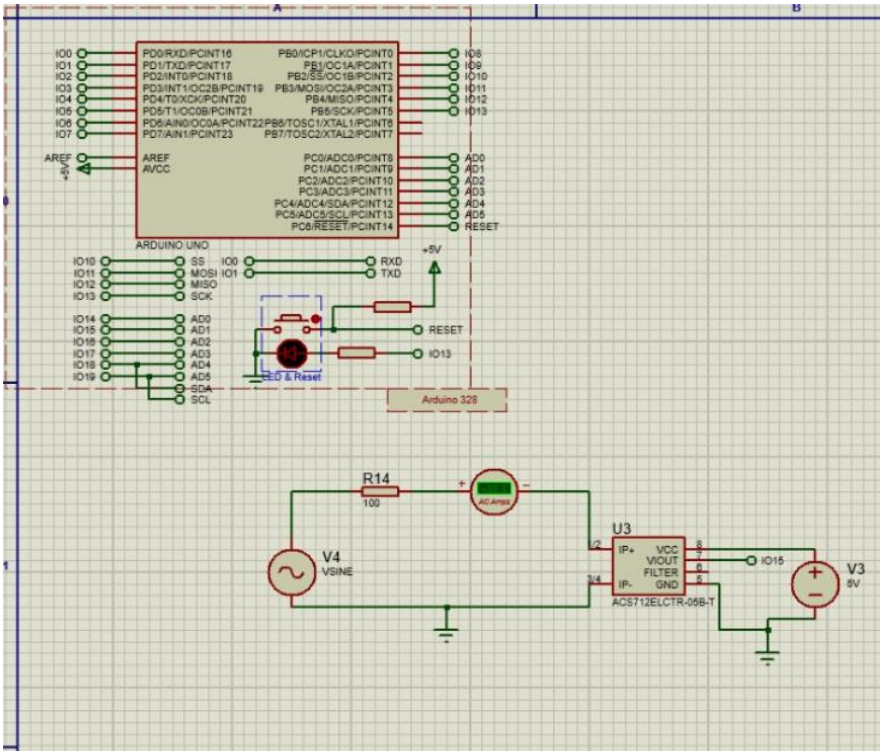
At first, we constructed the whole circuit on Proteus-

- Built the internal structure of ZMPT101B voltage sensor module as it's unavailable on proteus.
- Connected voltage source and load separately both on ACS712 and ZMPT101B module.
- Signal detected by Arduino in analog values and processed by the source code to give necessary output power.

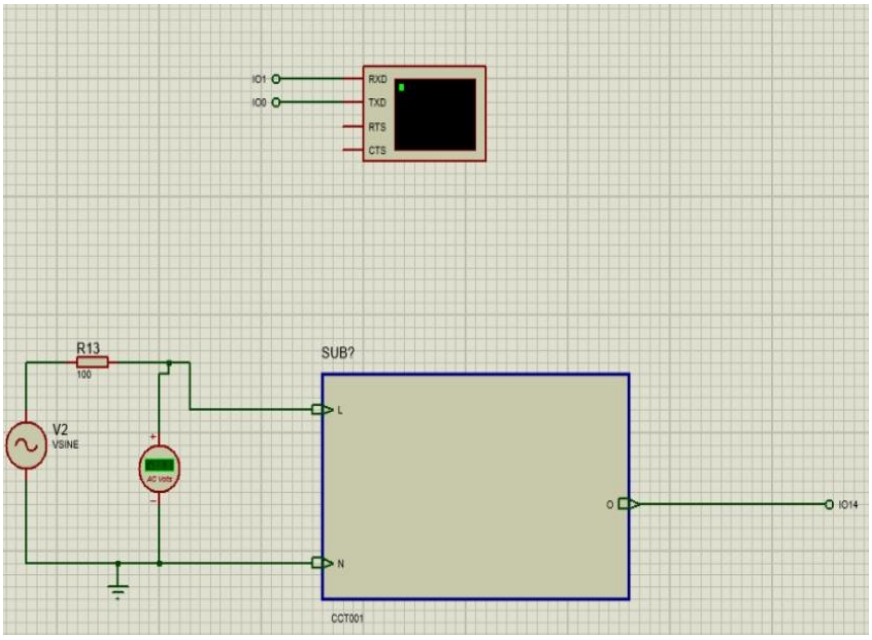
Circuit:



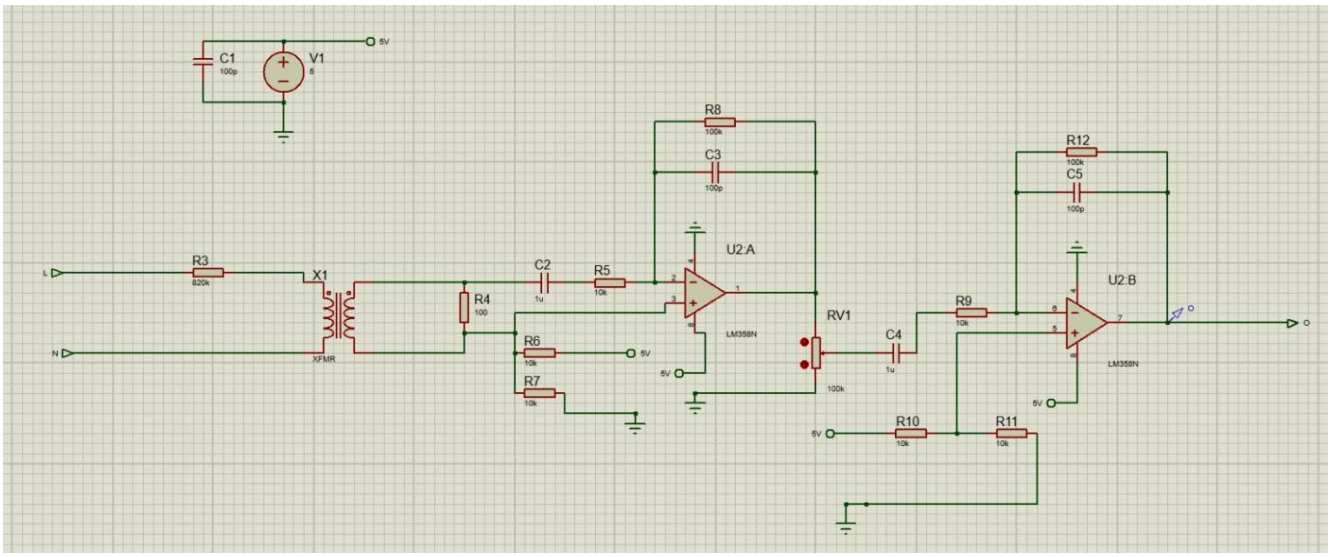
1.Smart Energy meter software module



2. Arduino and ACS712

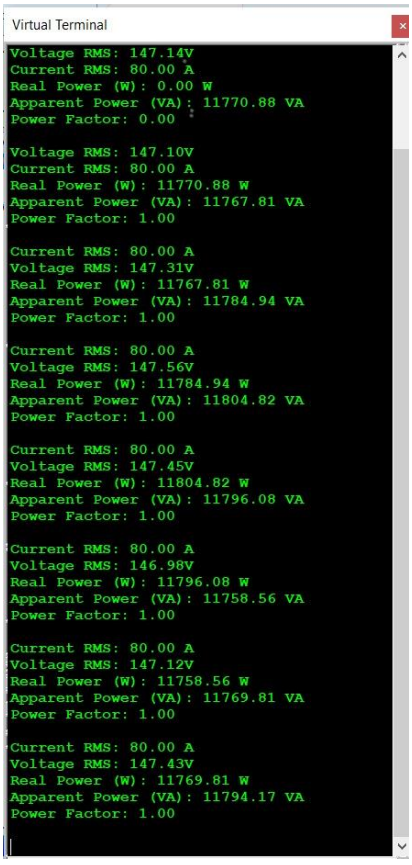


3. ZMPT101B and output block



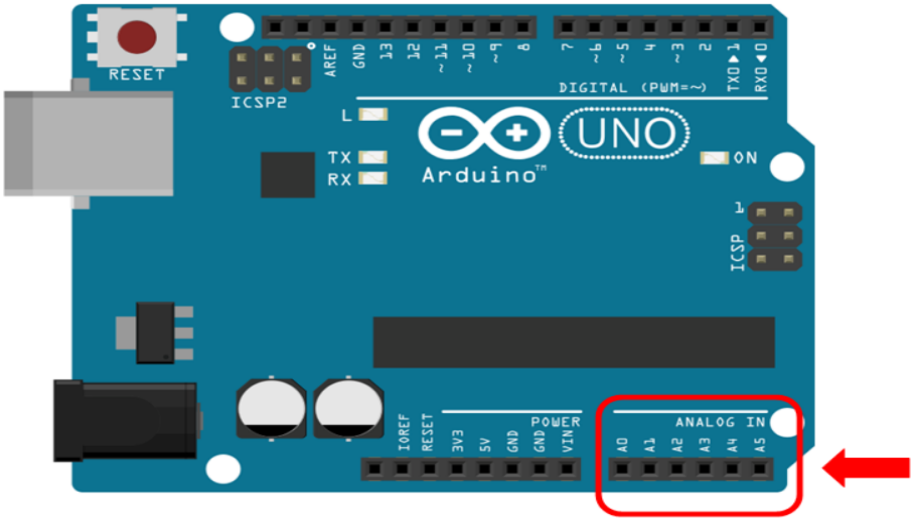
4.ZMPT101B Internal Structure

Output:



Components:

- **Arduino:**
  We will use Arduino to measure **Apparent Power** and **Real Power**. In order to measure **AC Power**, we need to use **2 separate sensors** which are Single Phase **AC Voltage Module and AC Current Module.** Each module can be fit into Arduino **analog input pin**. For Arduino UNO, there are **6 analog input pins** (A0-A5). Arduino NANO has 8 pins while Arduino MEGA has 16 input pins. The analog input pins will map input voltages between 0 and 5V into integer values between 0 and 1023 with resolution of 4.9mV per unit (5.00V / 1024 units).

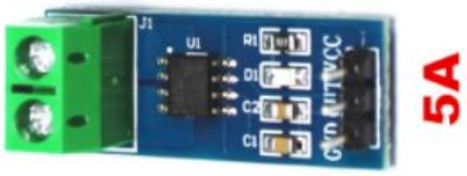


5. Arduino analog input pin

- Single Phase AC Voltage Module:
  This module is equipped with ZMPT101B high-precision voltage transformer and op amp circuit. It can measure AC voltage within 250V.

6.ZMPT101B Module

- **ACS712 Current Sensor Module**
  It is rated at **5A, 20A and 30A** which are suitable to most applications. The 5A module has the resolution of 185mV/ampere, 20A module has 100mV/ampere while 30A module has the resolution of 66mV/ampere. We will use 5A one in this project.



7.Current module

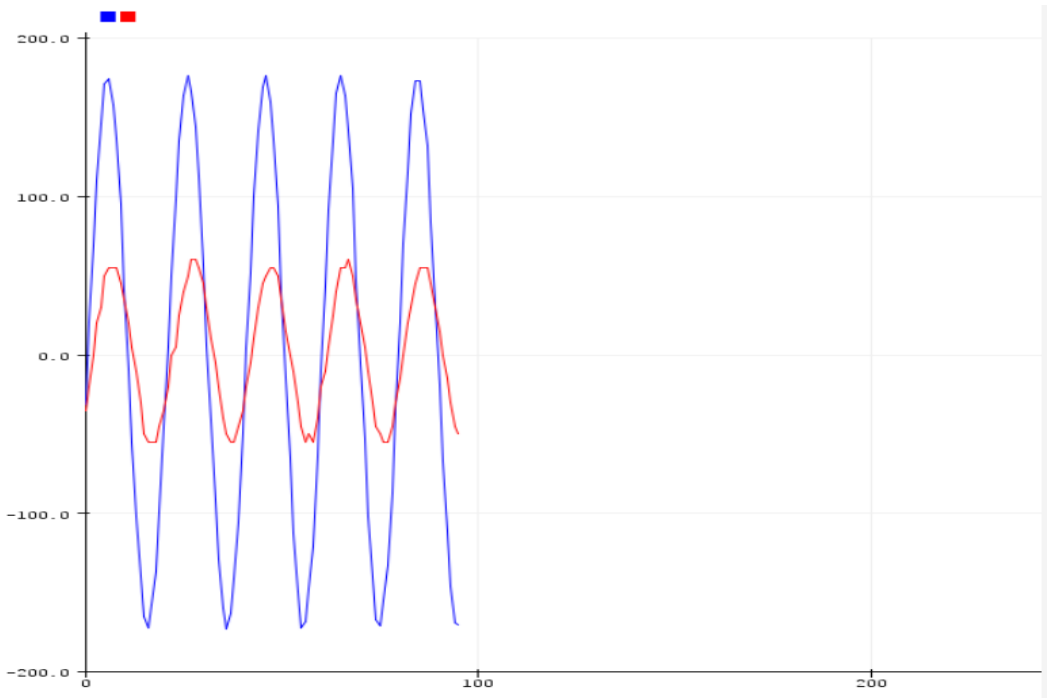
- **Relay:**
  A user current limit will be set. And if it crosses the set limit relay will automatically turn off which ultimately stops the flow of current.

8. Relay module

Functionality:

Signal detection:

Signals detected by Arduino are in **analog values**. Below are the signals being detected by the voltage (Blue) and current sensor (Red). We connected Voltage module in A2 pin of Arduino while Current module in A3 pin. The magnitude of voltage can be calibrated by AC voltage module using the on-board trimpot and compared with reference RMS volt meter. For a pure sine wave like this, the RMS value in voltmeter multiply by square root 2 is equal to peak voltage (magnitude from middle of oscillation). The magnitude of current wave (Red wave) is subject to applied load. The AC wave pattern might not be 100% smooth like voltage wave and it is subject to the load on how the current being drawn. The phase shift or phase difference between voltage and current also subject to applied load.



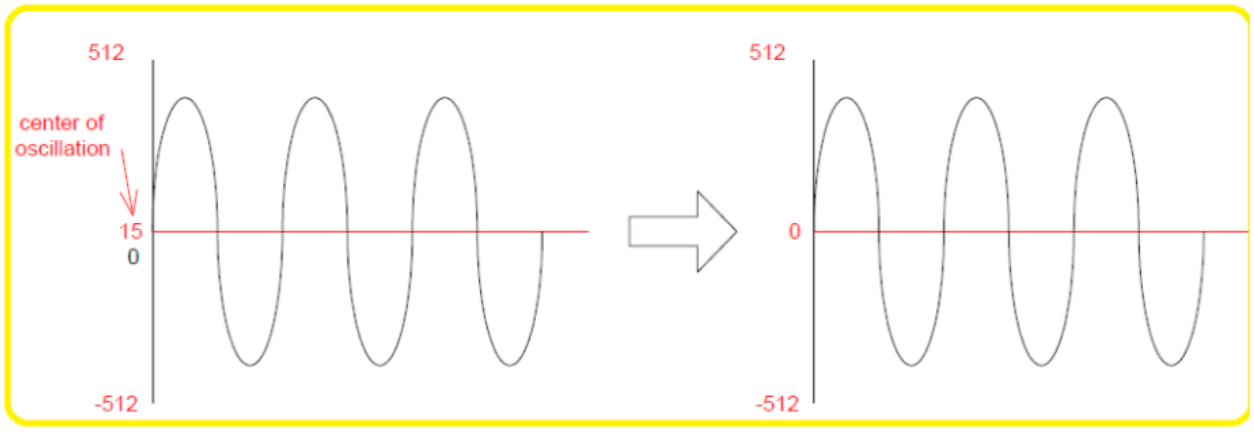
The middle point for both waves should be technically at 512 (analog value) and it fluctuates within 0 to 1023 by default. However, the code is amened so that the waves are oscillate at middle point 0 value between 512 and -512 for easier calculation later. Before going further, calibration is needed before the measurement as some modules might have different deviation error.

Signal Processing:

We need to determine the 2 AC Power values, the RMS AC Power and Active AC Power value. Before going further, let us begin with initial calibration for both sensors. We need to minimize any potential deviation error or inaccuracy as much as possible.

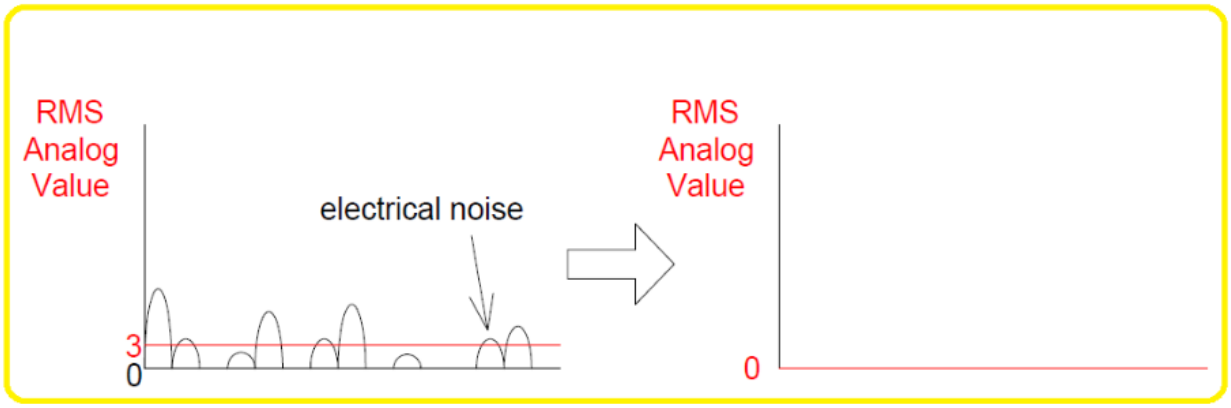
We need 2-time calibration for each module sensor. Both calibrations need to be done during no current & voltage measured. The first calibration is making sure when no voltage or current measured, it shows exactly at 0 point. It is an analog value calibration. Some modules might not show exactly at analog value 512 (so code is shifted to 0 point using Arduino code for easy understanding) when no value is detected. We need to add an offset value for this to adjust it back to origin when no value detected.

1st Offset / Calibration



The second calibration is to further eliminate false signal value during RMS calculation. Even after the first calibration is made, there are still some minor ghost or electrical noise even when no voltage and current are measured. We have to add another offset to make it to zero value at the final stage for display. This second calibration must be done only after the first calibration take into effect. Both calibrations can be done manually (the harder way) or automatically by pressing the SELECT Button in the LCD Display Shield and wait for about 5 seconds.

2nd Offset / Calibration

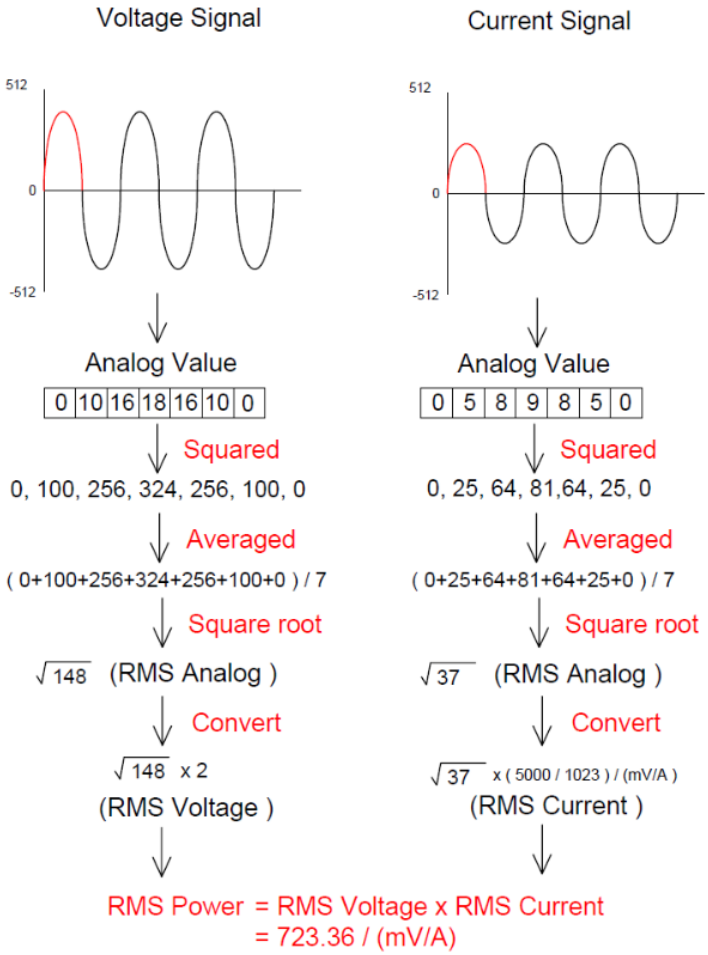


The Root Mean Square Power Calculation

The **RMS AC Power** is the product of **RMS AC Voltage** and **RMS AC Current** values. The RMS AC Voltage and RMS AC Current are calculated separately. By finding RMS AC Power, RMS Voltage and RMS current are indirectly to be found.

Our code is designed to calculate or display a value which is derived from averaging 1000 samples per set. We have set each sample is recorded every 1 milli second (0.001 second). In other words the total complete set is equivalent to 50 waves with each wave is divided into 20 sections or readings (for 50Hz). Technically it should last 1 second for 1 set of reading. It is also suitable for 60Hz system as the measurement time for 1 set of reading is 1 second.

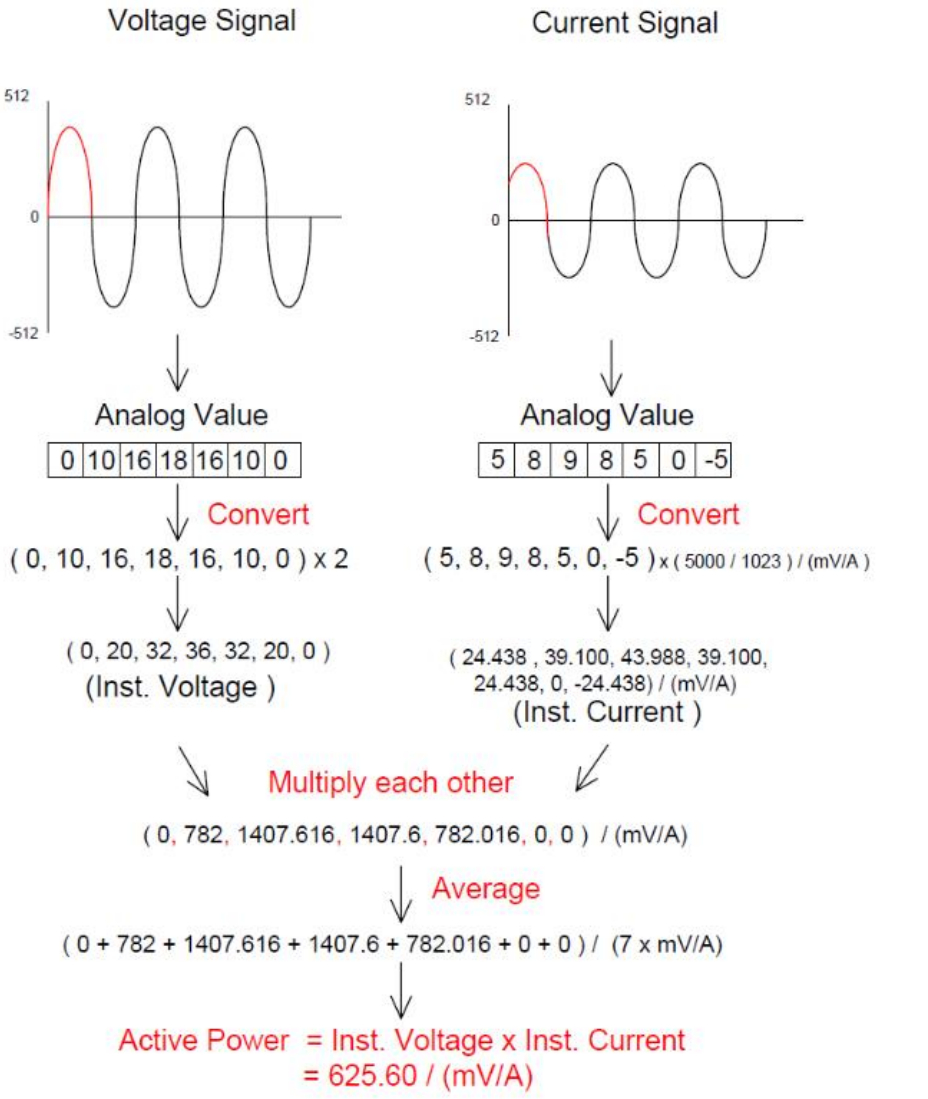
Each single sample analog value is being squared initially and once the 1000 sample values are accumulated, the average value from the 1000 samples is then being square-rooted in order to come out the RMS analog value (for 1 sensor). We then convert the RMS analog value into measured voltage or current value. The similar calculation is separately done for the other sensor and multiply both RMS values to become RMS AC Power. Below is an example how the code works. A quarter of wave is taken as an example.



9. Calculation of Power

The Instantaneous Power Calculation

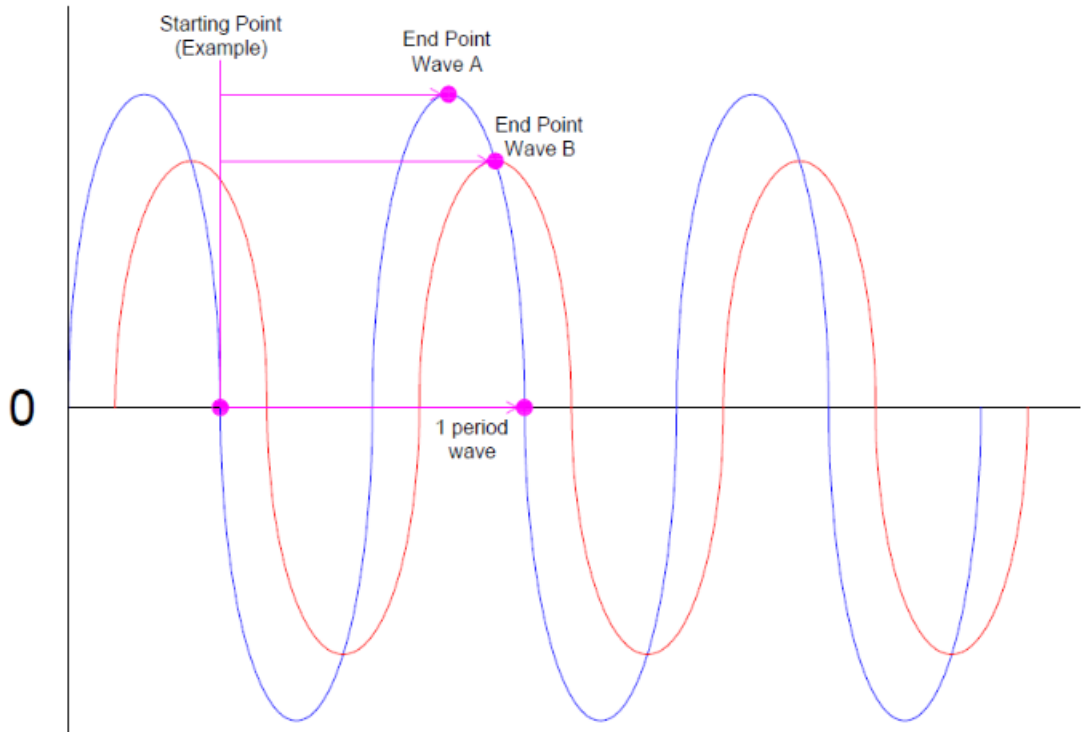
Active or Real Power is simpler compared to RMS Power. It is the average value of all the multiplications of instantaneous voltage and instantaneous current value. Both voltage and current analog values are first converted into measured voltage and current value. The voltage is then multiplied by its instantaneous current value and become a sample reading. An example below is listed:



Similar to RMS power method, the Active Power is also derived from averaging 1000 samples per set. Each sample is recorded every milli second, in other words the total complete set is equivalent to 50 waves with each wave is divided into 20 sections or readings (for 50Hz).

Measuring Power factor and Phase angle

The code uses AnalogInput1 (Wave A) as reference. It first determines the entry point to start recording time. The point is where the wave crosses over or below 0 value for voltage sensor (wave A). The starting time are the same for all 3 measurement parameters (wave A, wave B and frequency of wave A).



When Wave A more than 0, it keeps checking and recording the current time. The time recording is based on whether the recording analog value larger than previous measured value. If at the peak where current recording value no longer larger than previous value, recording time stops and it will be the last recording time. The same applies to wave B which separate time recording and stops at peak value. The time taken from starting point to peak value for wave A is estimate at 75% of the whole period wave. The time difference taken between wave A and wave B to reach peak is actually the phase shift time which phase angle can be calculated.

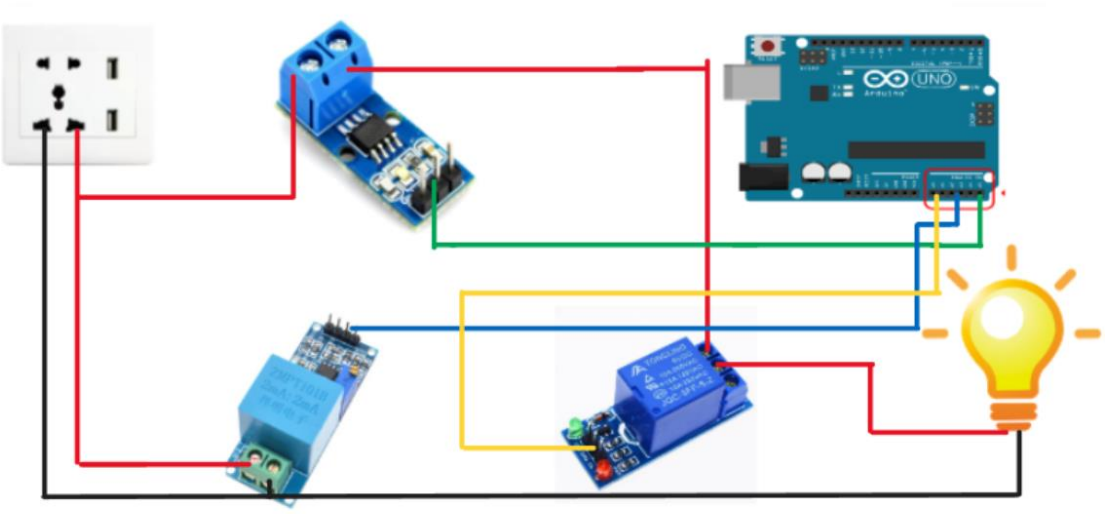
When the Wave A back to 0, the time recording for period or frequency is stopped. Period is the time difference between starting point and end point. At the same time, the current time is set as starting time for the next cycle of reading set and the process repeated until 50 cycles (for 50Hz) or 60 cycles (for 60 Hz).

LCD Display Shield

This is a shield that allows the output value of Arduino board to be displayed on the screen. Since it is a shield, we can just stack it on Arduino board without the need of extra wiring for the LCD Display.

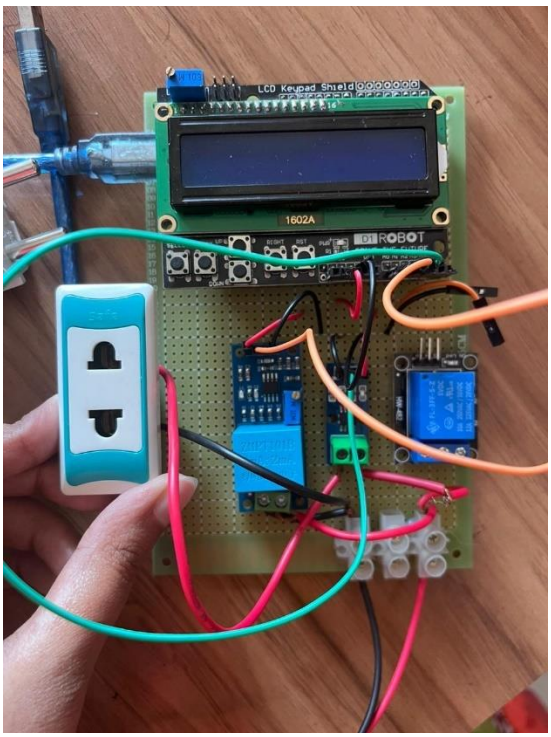
10. LCD Display

Hardware Connection:



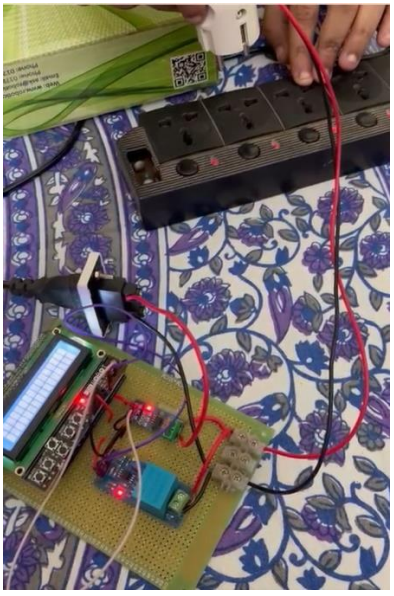
11. Hardware Setup

Hardware Module:



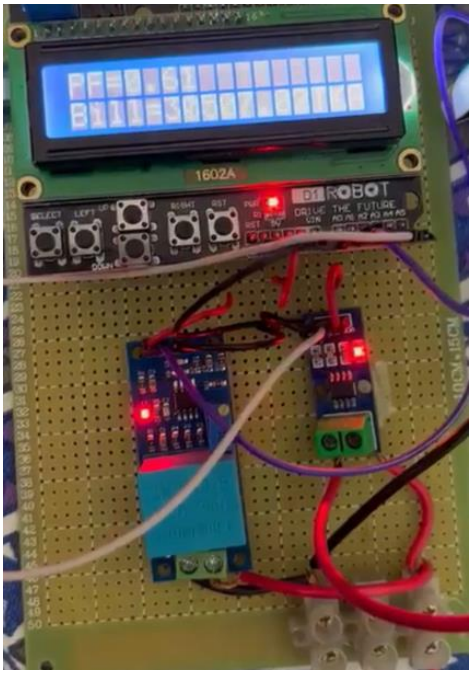
12. Hardware implementation

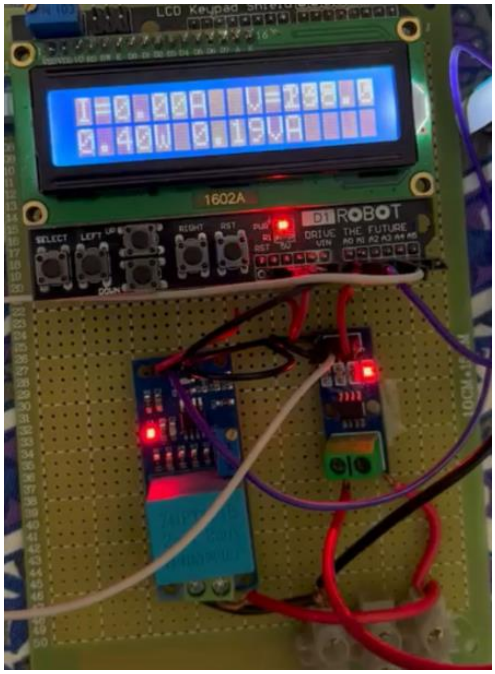
1. Taking Table fan as load:



13. Setup with table fan

Output:

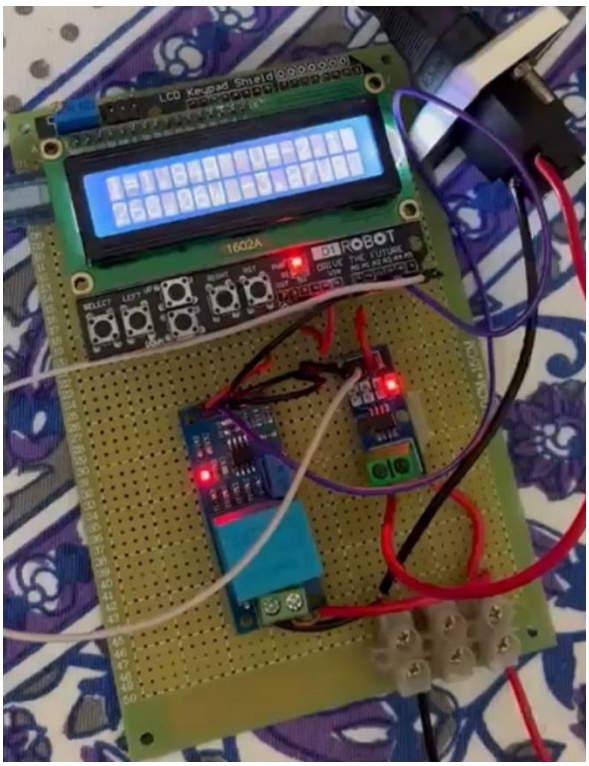




14. Voltage, Current, PF and bill

2. Taking Hair dryer as load:

Output:



Future Work:

We can implement many features like-

- Give over current protection. If any device is taking huge current then there will be circuit breaker and sensors or relays which will automatically turn off the current flow.
- Having a password protected device. Therefore, if the necessary password is entered, the circuit breaker will activate, allowing the device to function; otherwise, the gadget won't be able to draw electricity from the socket to which the energy meter is connected.
- This is a modular design. It means we can add as many features we want in the future.

In this way, more works can be done in this project.

Acknowledgement:

We would like to thank all our teachers especially our supervisor for guiding us through this project.

Conclusion:

This smart energy-measuring device may educate the general population about the cost of power and assist them in determining how much they consume on a daily basis. Power bills are now paid in advance. As a result, having a device that can calculate the needed cost of each electrical item will be tremendously convenient for us. This is critical for both residential and industrial users to efficiently control their energy bills. Users can change their behavior or equipment to decrease energy waste, which helps the environment and saves money.

Codes:

```
/* 0- General */

 int decimalPrecision = 2;

/* 1- AC Voltage Measurement */

        int VoltageAnalogInputPin = A1;              // Which pin to measure voltage Value

     float voltageSampleRead  = 0;               /* to read the value of a sample*/

     float voltageLastSample  = 0;               /* to count time for each sample. Technically 1 milli second
1 sample is taken */

     float voltageSampleSum   = 0;               /* accumulation of sample readings */

     float voltageSampleCount = 0;               /* to count number of sample. */

     float voltageMean ;

     float RMSVoltageMean ;


         /*1.1 Offset AC Voltage */


         int voltageOffsetRead = 0;              /* to change the mode for offset */

         float voltageOffset1 = 0;               // to Offset deviation and accuracy. Offset any fake current
when no current operates.

                                                 // Offset will automatically callibrate when SELECT Button on the LCD
Display Shield is pressed.

                                                 // If you do not have LCD Display Shield, look into serial monitor to add
or minus the value manually and key in here.

         float voltageOffset2 = 0;               // to offset value due to calculation error from squared and
square root.

         float voltageSampleSumOffset =0;        /* accumulation of sample readings for offset */

         float offsetVoltageMean = 0;            /* to calculate the average value from all samples for offset,
in analog values*/

         float voltageOffsetLastSample = 0;      /* to count time for each sample for offset purpose. */

         float voltageOffsetSampleCount = 0;     /* to count number of sample for offset. */
```

```
/* 2- AC Current Measurement */


int CurrentAnalogInputPin = A3;              // Which pin to measure Current Value

float mVperAmpValue = 185;                   // If using ACS712 current module : for 5A module key in 185, for 20A module key in 100, for 30A module key in 66

                                             // If using "Hall-Effect" Current Transformer, key in value using this formula: mVperAmp = maximum voltage range (in milli volt) / current rating of CT

                                             /* For example, a 20A Hall-Effect Current Transformer rated at 20A, 2.5V +/- 0.625V, mVperAmp will be 625 mV / 20A = 31.25mV/A */

float currentSampleRead  = 0;                /* to read the value of a sample*/

float currentLastSample  = 0;                /* to count time for each sample. Technically 1 milli second 1 sample is taken */

float currentSampleSum   = 0;                /* accumulation of sample readings */

float currentSampleCount = 0;                /* to count number of sample. */

float currentMean ;                          /* to calculate the average value from all samples*/

float RMSCurrentMean =0 ;                    /* square roof of currentMean*/

float FinalRMSCurrent ;                      /* the final RMS current reading*/

/*2.1 Offset AC Current */


int currentOffsetRead = 0;                   /* to change the mode for offset */

float currentOffset1 = 0;                    // to Offset deviation and accuracy. Offset any fake current when no current operates.

                                             // Offset will automatically callibrate when SELECT Button on the LCD Display Shield is pressed.

                                             // If you do not have LCD Display Shield, look into serial monitor to add or minus the value manually and key in here.

                                             // 26 means add 26 to all analog value measured

float currentOffset2 = 0;                    // to offset value due to calculation error from squared and square root.

float currentSampleSumOffset = 0;            /* accumulation of sample readings for offset */

float offsetCurrentMean = 0;                 /* to calculate the average value from all samples for offset, in analog values*/
```

```
        float currentOffsetLastSample = 0;        /* to count time for each sample for offset purpose. */
        float currentOffsetSampleCount = 0;       /* to count number of sample for offset. */
  /* 3- AC Power Measurement */

      float sampleCurrent1 ;                    /* use to calculate current*/
      float sampleCurrent2 ;                    /* use to calculate current*/
      float sampleCurrent3 ;                    /* use to calculate current*/
      float apparentPower;                      /* the apparent power reading (VA) */
      float realPower = 0;                     /* the real power reading (W) */
      float powerSampleRead  = 0;               /* to read the current X voltage value of a sample*/
      float powerLastSample   = 0;              /* to count time for each sample. Technically 1 milli second
1 sample is taken */
      float powerSampleCount  = 0;              /* to count number of sample. */
      float powerSampleSum    = 0;              /* accumulation of sample readings */
      float powerFactor = 0;                    /* to display power factor value*/
      float r=0;
      float t=0;
      float t_hour=0;
      float bill=0;
      float unit_price=380;
 /*3.1 Offset AC Power */

        int powerOffsetRead = 0;                /* to change the mode for offset */
        float powerOffset = 0;                  // to Offset deviation and accuracy. Offset any fake current
when no current operates.
                                                // Offset will automatically callibrate when SELECT Button on the LCD
Display Shield is pressed.
                                                // If you do not have LCD Display Shield, look into serial monitor to add
or minus the value manually and key in here.
        float powerOffsetLastSample = 0;        /* to count time for each sample for offset purpose. */
```

```
    float powerOffsetSampleCount = 0;          /* to count number of sample for offset. */


    /* 4 - LCD Display  */

    #include<LiquidCrystal.h>                  /* Load the liquid Crystal Library (by default already built-it with arduino solftware)*/
    LiquidCrystal LCD(8,9,4,5,6,7);            /* Creating the LiquidCrystal object named LCD. The pin may be varies based on LCD module that you use*/
    unsigned long startMillisLCD;              /* start counting time for LCD Display */
    unsigned long currentMillisLCD;            /* current counting time for LCD Display */
    const unsigned long periodLCD = 1000;      // refresh every X seconds (in seconds) in LED Display. Default 1000 = 1 second
    int page = 1;                              /* flip page to display values*/

void setup()                                   /*codes to run once */

{

    /* 0- General */

    Serial.begin(9600);                        /* to display readings in Serial Monitor at 9600 baud rates */

    /* 4 - LCD Display  */

    LCD.begin(16,2);                           /* Tell Arduino that our LCD has 16 columns and 2 rows*/
    LCD.setCursor(0,0);                        /* Set LCD to start with upper left corner of display*/
    startMillisLCD = millis();                 /* Start counting time for LCD display*/

}
```

```
void loop()                                                    /*codes to run again and again */
{

      /* 0- General */


            /* 0.1- Button Function */


            int buttonRead;
            buttonRead = analogRead (0);                                    // Read analog pin A0. Pin A0
automatically assigned for LCD Display Button function (cannot be changed)


            /*Right button is pressed */
            if (buttonRead < 60)
            {   LCD.setCursor(0,0); LCD.print ("PRESS <SELECT>   "); }


            /* Up button is pressed */
            else if (buttonRead < 200)
            {
            page = 1;
            }


            /* Down button is pressed */
            else if (buttonRead < 400)
            {
            page = 2;
            }


            /* Left button is pressed */
```

```
        else if (buttonRead < 600)
        {   LCD.setCursor(0,0); LCD.print ("PRESS <SELECT>   "); }

        /* Select button is pressed */
        else if (buttonRead < 800)
        {
        currentOffsetRead = 1;                                  // to activate offset for current
        voltageOffsetRead = 1;                                  // to activate offset for voltage
        powerOffsetRead = 1;                                    // to activate offset for power
        LCD.setCursor(0,0);                                     /* set display words starting at upper left
corner*/
        LCD.print ("INITIALIZING..... ");
        LCD.setCursor(0,1);                                     /* set display words starting at lower left
corner*/
        LCD.print ("WAIT 5 SEC ..... ");
        }
    /* 1- AC Voltage Measurement */

    if(millis() >= voltageLastSample + 1 )                      /* every 1 milli second taking 1
reading */
      {
        voltageSampleRead = 2*(analogRead(VoltageAnalogInputPin)- 512) + voltageOffset1;    /* read
the sample value */
        voltageSampleSumOffset = voltageSampleSumOffset + voltageSampleRead;        /* values
accumulate for offset purpose every milli second */

        voltageSampleSum = voltageSampleSum + sq(voltageSampleRead) ;               /* accumulate
value with older sample readings*/
        voltageSampleCount = voltageSampleCount + 1;                /* to move on to the next
following count */
        voltageLastSample = millis() ;                              /* to reset the time again so that next
cycle can start again*/
```

```
    }
   if(voltageSampleCount == 1000)                                  /* after 1000 count or 1000 milli seconds (1 second), do the calculation and display value*/
    {
     offsetVoltageMean = voltageSampleSumOffset/voltageSampleCount;          /* average the offset reading*/

     voltageMean = voltageSampleSum/voltageSampleCount;                  /* calculate average value of all sample readings taken*/
     RMSVoltageMean = sqrt(voltageMean)+ voltageOffset2;                 /* square root of the average value*/
     Serial.print(" The Voltage RMS value is: ");
     Serial.print(RMSVoltageMean,decimalPrecision);
     Serial.println(" V ");
     voltageSampleSum =0;                                        /* to reset accumulate sample values for the next cycle */
     voltageSampleCount=0;                                       /* to reset number of sample for the next cycle */
     voltageSampleSumOffset=0;
    }
      /* 1.1 - Offset AC Voltage */

      if(voltageOffsetRead == 1)                                 /* Run this code when button SELECT is pressed */
       {
        voltageOffset1 = 0;
        if(millis()>= voltageOffsetLastSample + 1)                 /* keep countng time for offset1*/
         {
          voltageOffsetSampleCount = voltageOffsetSampleCount + 1;          /* 1 milli second add 1 count*/
```

```
            voltageOffsetLastSample = millis();                    /* to reset the time again so that
next cycle can start again*/
          }
         if(voltageOffsetSampleCount == 2000)                    /* after 2 seconds, run this
codes.  */
          {
            voltageOffset1 = -1*(offsetVoltageMean);              /* set the offset values */
            voltageOffsetRead = 2;                               /* go for second offset Settings */
            voltageOffsetSampleCount = 0;                        /* to reset the time again so that
next cycle can start again */
          }
        }

      if(voltageOffsetRead == 2)                                 /* Run this code after first offset done
*/
       {
         voltageOffset2 = 0;                                   /* set back currentOffset2 as default*/
         if(millis()>= voltageOffsetLastSample + 1)            /* keep countng time for
offset2*/
          {
            voltageOffsetSampleCount = voltageOffsetSampleCount + 1;
            voltageOffsetLastSample = millis();
          }
         if(voltageOffsetSampleCount == 2000)                    /* after 2 seconds, run this
codes.  */
          {
            voltageOffset2 = - RMSVoltageMean;                   /* set the offset values */
            voltageOffsetRead = 0;                               /* change the offset mode to original,
wait until the button is pressed again */
            voltageOffsetSampleCount = 0;                        /* to reset the time again so that
next cycle can start again */
          }
```

```
        }
    /* 2- AC Current Measurement */

     if(millis() >= currentLastSample + 1)                    /* every 1 milli second taking 1 reading */
      {
        currentSampleRead = analogRead(CurrentAnalogInputPin)-512 + currentOffset1;      /* read the sample value */

        currentSampleSumOffset = currentSampleSumOffset + currentSampleRead;          /* accumulate offset value */

        currentSampleSum = currentSampleSum + sq(currentSampleRead) ;              /* accumulate value with older sample readings*/
        currentSampleCount = currentSampleCount + 1;                    /* to move on to the next following count */
        currentLastSample = millis();                          /* to reset the time again so that next cycle can start again*/
      }

     if(currentSampleCount == 1000)                          /* after 1000 count or 1000 milli seconds (1 second), do the calculation and display value*/
      {
        offsetCurrentMean = currentSampleSumOffset/currentSampleCount;              /* average offset value*/

        currentMean = currentSampleSum/currentSampleCount;                  /* calculate average value of all sample readings taken*/
        RMSCurrentMean = sqrt(currentMean)+currentOffset2 ;                 /* square root of the average value*/
        FinalRMSCurrent = (((RMSCurrentMean /1024) *5000) /mVperAmpValue);          /* calculate the final RMS current*/
        Serial.print(" The Current RMS value is: ");
```

```
        Serial.print(FinalRMSCurrent,decimalPrecision);
        Serial.println(" A ");
        currentSampleSum =0;                                    /* to reset accumulate sample values
for the next cycle */
        currentSampleCount=0;                                   /* to reset number of sample for the
next cycle */
        currentSampleSumOffset=0;                               /* to reset accumulate offset value
for the next cycle*/
      }
      /* 2.1 - Offset AC Current */

        if(currentOffsetRead == 1)                              /* Run this code when button SELECT
is pressed */
          {
            currentOffset1 = 0;                                 /* set currentOffset back to default
value*/
            if(millis()>= currentOffsetLastSample + 1)          /* keep countng time for
offset1*/
             {
               currentOffsetSampleCount = currentOffsetSampleCount + 1;
               currentOffsetLastSample = millis();
             }
            if(currentOffsetSampleCount == 2000)                /* after 2 seconds, run this
codes.  */
             {
               currentOffset1 = - offsetCurrentMean;            /* set the offset values */
               currentOffsetRead = 2;                           /* go for second offset Settings */
               currentOffsetSampleCount = 0;                    /* to reset the time again so that
next cycle can start again */
             }
          }
```

```
        if(currentOffsetRead == 2)                                  /* Run this code after first offset done
*/
          {
            currentOffset2 = 0;                                    /* set back currentOffset2 as default*/
            if(millis()>= currentOffsetLastSample + 1)                    /* keep countng time for
offset2*/
             {
               currentOffsetSampleCount = currentOffsetSampleCount + 1;
               currentOffsetLastSample = millis();
             }

            if(currentOffsetSampleCount == 2000)                       /* after 2 seconds, run this
codes.  */
             {
               currentOffset2 = - RMSCurrentMean;                      /* set the offset values */
               currentOffsetRead = 0;                                 /* change the offset mode to original,
wait until the button is pressed again */
               currentOffsetSampleCount = 0;                          /* to reset the time again so that
next cycle can start again */
             }
          }

    /* 3- AC Power with Direction */

    if(millis() >= powerLastSample + 1)                                /* every 1 milli second taking 1
reading */
      {
        sampleCurrent1 = analogRead(CurrentAnalogInputPin)-512+ currentOffset1;
        sampleCurrent2 = (sampleCurrent1/1024)*5000;
        sampleCurrent3 = sampleCurrent2/mVperAmpValue;
        voltageSampleRead = 2*(analogRead(VoltageAnalogInputPin)- 512)+ voltageOffset1 ;
```

```
        powerSampleRead = voltageSampleRead * sampleCurrent3 ;                    /* real power
sample value */

        powerSampleSum = powerSampleSum + powerSampleRead ;                       /* accumulate
value with older sample readings*/

        powerSampleCount = powerSampleCount + 1;                                  /* to move on to the next
following count */

        powerLastSample = millis();                                               /* to reset the time again so that next
cycle can start again*/

       }


     if(powerSampleCount == 1000)                                                 /* after 1000 count or 1000 milli
seconds (1 second), do the calculation and display value*/

       {

        realPower = ((powerSampleSum/powerSampleCount)+ powerOffset) ;            /* calculate
average value of all sample readings */

        Serial.print(" The Real Power (W) is: ");

        Serial.print(realPower);

        Serial.println(" W ");

        apparentPower= FinalRMSCurrent*RMSVoltageMean;                            /*Apparent power do
not need to recount as RMS current and RMS voltage values available*/

        Serial.print(" The Apparent Power (VA) is: ");

        Serial.print(apparentPower,decimalPrecision);

        Serial.println(" VA ");

        powerFactor = realPower/apparentPower;

        if(powerFactor >1 || powerFactor<0)

        {

          powerFactor = 0;

        }

        Serial.print(" The Power Factor is: ");

        Serial.println(powerFactor,decimalPrecision);

        powerSampleSum =0;                                                        /* to reset accumulate sample values
for the next cycle */
```

```
        powerSampleCount=0;                                    /* to reset number of sample for the
next cycle */
      }
     /* 3.1 - Offset AC Power */

       if(powerOffsetRead == 1)                                /* Run this code after first offset done
*/
        {
          powerOffset = 0;                                   /* set back currentOffset2 as default*/
          if(millis()>= powerOffsetLastSample + 1)                    /* keep countng time for
offset2*/
           {
            powerOffsetSampleCount = powerOffsetSampleCount + 1;
            powerOffsetLastSample = millis();
           }
          if(powerOffsetSampleCount == 5000)                     /* after 5 seconds, run this
codes.  */
           {
            powerOffset = -realPower;
            powerOffsetRead = 0;                               /* change the offset mode to original,
wait until the button is pressed again */
            powerOffsetSampleCount = 0;                        /* to reset the time again so that
next cycle can start again */
           }
        }
  /* 4 - LCD Display  */

    currentMillisLCD = millis();                               /* Set current counting time */
    if (currentMillisLCD - startMillisLCD >= periodLCD && page ==1)          /* for every x
seconds, run the codes below*/
     {
```

```
        LCD.setCursor(0,0);                                   /* Set cursor to first colum 0 and second
row 1  */

        LCD.print("I=");

        LCD.print(FinalRMSCurrent,decimalPrecision);                    /* display current value in
LCD in first row  */

        LCD.print("A  ");

        LCD.print("V=");

        LCD.print(RMSVoltageMean,decimalPrecision);                     /* display current value in
LCD in first row  */

        LCD.print("V     ");

        LCD.setCursor(0,1);

        LCD.print(realPower,decimalPrecision);

        LCD.print("W ");

        LCD.print(apparentPower,decimalPrecision);                      /* display current value in
LCD in first row  */

        LCD.print("VA    ");

        startMillisLCD = currentMillisLCD ;                   /* Set the starting point again for
next counting time */

       }

        r= realPower/1000;

        t= millis();

        t_hour= (t/ (3600*1000U));

        bill= r*t_hour*unit_price;


     if( currentMillisLCD - startMillisLCD >= periodLCD && page ==2)

      {

        LCD.setCursor(0,0);                                   /* Set cursor to first colum 0 and second
row 1  */

        LCD.print("PF=");

        LCD.print(powerFactor,decimalPrecision);

        LCD.print("          ");
```

```
        LCD.setCursor(0,1);
        //LCD.print("              ");
        LCD.print("Bill=");
        LCD.print(bill);
        LCD.print("TK");
        LCD.print("              ");
        startMillisLCD = currentMillisLCD ;                    /* Set the starting point again for
next counting time */
      }



}
```