A PRACTICAL TRAINING REPORT ON A MODERN APPORACH TO MEASURE ELECTRICAL PARAMETER

Submitted for the partial fulfillment of Degree

Bachelor of Technology

ELECTRICAL AND ELECTRONICS ENGINEERING

Submitted by: -
VIJAY SINGH
12EBKEX061

Submitted to:-
Dr. L Solanki
Principal (Academics)

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
B.K. BIRLA INSTITUTE OF ENGINEERING & TECHNOLOGY, PILANI (RAJ)

(Affiliated to Rajasthan Technical University, Kota)

2015-16

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and indebtedness to **MR. Ajeet Dhakar** for allowing me to undergo the summer training of 45 days at CENTRAL ELECTRONICS ENGINEERING RESEARCH INSTITUTE PILANI.

I am grateful to **MR. Vinod Verma** for the help provided in completion of the training, which was assigned to me.

I am also thankful to **Dr. P S Bhatnagar** (Director BKBIET,Pilani) for his true help & providing the resources on completion of training. Last but not least, I pay my sincere thanks and gratitude to **Dr. L Solanki** (Principal Academics BKBIET,Pilani) for his valuable suggestions & encouragement.

I also like to thank **Mr. Santosh Jangid** (Assistant Professor BKBIET, Pilani) and **Mr. Rajesh Singh Shekhawat** (Assistant Professor BKBIET, Pilani) for giving the support and suggestions.

I would like to take this opportunity to thanks **Mr**. **Satish Kumar Rai** for his guidance and motivation which helps in doing this work.

Table of Contents

TABLE OF FIGURES

CEERI

Central Electronics Engineering Research Institute (CEERI), located at Pilani, Rajasthan and Chennai, Tamil Nadu is a pioneer research institute in India and a constituent laboratory of Council of Scientific and Industrial Research (CSIR India), New Delhi. It was established in 1953 for advanced research and development in the field of Electronics.[1]

Since its inception, it has been working for the growth of electronics in the country and has established the required infrastructure and well experienced manpower for undertaking R&D in the following major areas :

Electronics System : Areas of Research

- Agri-Electronics
- Embedded Systems
- Digital Systems
- Power Electronics

Electron Tubes : Areas of Research

- Gyrotron
- Klystron
- Magnetrons
- Plasma Devices
- Traveling Wave Tubes

Semiconductor : Areas of Research

- Hybrid Mircrocircuits
- IC Design
- MEMS and Microsensors
- Sensors and Nanotechnology
- Photonics and Optoelectronics
- Semiconductor Materials and Technology

Abstract

Measurement of electrical parameter is necessary and important for every electrical system .before digital techniques it was done using the mechanical techniques which need more power now with the modern technology it can be done using digital technique by doing signal conditioning and adding microcontroller.

In this report I discuss about the measurement of voltage, current, frequency and energy using ARDUINO and display them on the LCD screen. Voltage is measured using the 10 bit ADC on the microcontroller and frequency is sensed by the digital pin of ARDUINO for the current CT TALENA AC1020and ACS 710 is used. For the calculation of time RTC module is used and ARDUINO IDE is used for the programming platform.

WEEK 1

1 Introduction

1.1 Introduction to electrical system:

An electric power system is a network of electrical components used to supply, transmit and use electric power. An example of an electric power system is the network that supplies a region's homes and industry with power. For the control and use we need the correct reading of electrical parameter

1.1.1 Voltage

Voltage is equal to the work done per unit of charge against a static electric field to move the charge between two points. In simple way it can be said that it is the difference of electrical potential energy between the two points on the electrical element. Measured in units of potential volts, or joules per coulomb.

The measurement of voltage is done using voltmeter. One volt is defined as the difference in electric potential between two points of a conducting wire when an electric current of one ampere dissipates one watt of power between those points.

1.1.2 Electrical current

It is the flow of charge when potential difference is present between two points on the electrical element can be defined as the rate of flow of charge. Unit of current is ampere (A). one ampere is said as the one coulomb of charge flow through a surface in one second.

The current always flow in a closed loop. The measurement of current is done using ammeter. The flow of current can be direct current of alternating current direct current is that which does not change the magnitude over time where as alternate current changes the direction of flow on time interval, both type of current is used in electrical system.

1.1.3 Frequency

It can simply define as the number of cycles in one second. In electrical system in alternating current the direction of flow changes from positive to negative hence oscillation can be measured in frequency. Unit of frequency is HERTZ (HZ).

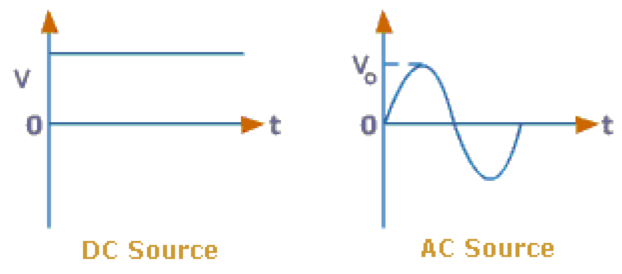


Figure 1.1 dc and ac source wave forms

1.1.4 ENERGY

The rate at which electric energy is transferred by an electric circuit is called power. Power is an important electrical quantity and everything in our world today depends on having the power to keep them running. It is necessary for any electrical system to know that how much power is generated and uses by load. For DC and pure resistive AC system power is multiple of voltage and current, and energy can be calculated using integrating power with time. Unit of power is watt and unit of energy is joules. Here energy is calculated in KWH it is a standard unit for energy used in electrical energy meter. For 1 KWH= 1 UNIT. The total cost of electrical energy consumed is calculated by multiplying the number of units with the rate at which electricity is provided .

Total cost = number of units*rate of electricity per unit.

1.2 Arduino

1.2.1 What is Arduino?

It is a open source platform where one can find the hardware, software and can modify them according to the requirements.

1.2.2 Hardware

The hardware can also be termed as boards.

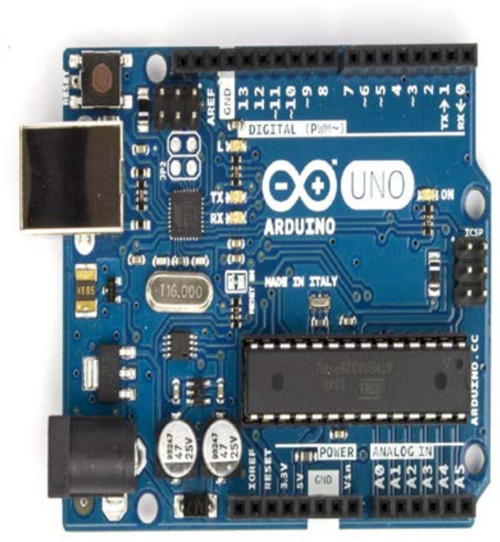


Figure 1.2 Arduino board R3 front

Figure 1.3 Arduino R3 back

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by boot loader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
Length	68.6 mm

1.2.3 Software

Arduino release different version of software for windows and mac OS.C programming is used to write the code.

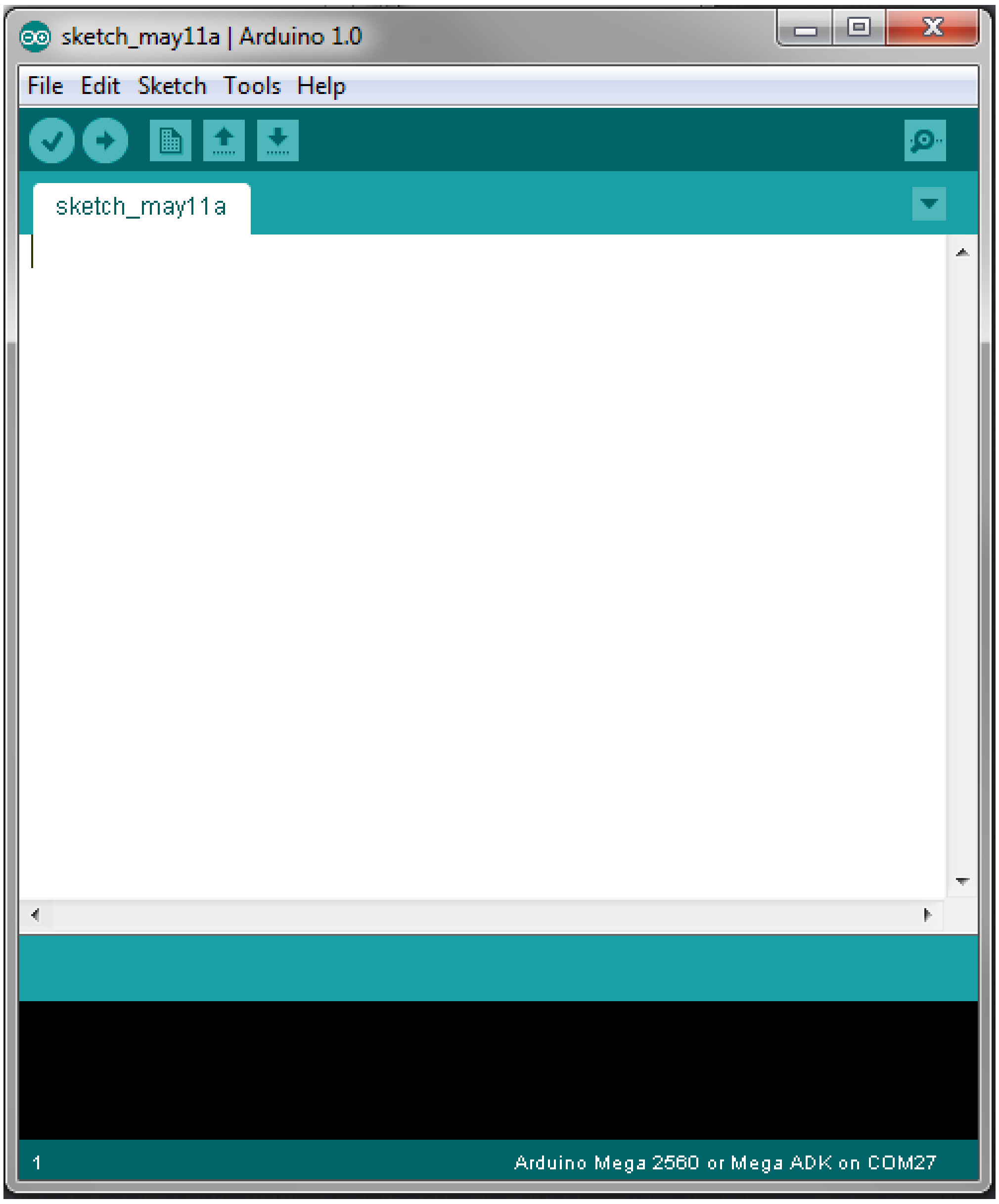

sketch_may11a | Arduino 1.0
File Edit Sketch Tools Help
sketch_may11a
1
Arduino Mega 2560 or Mega ADK on COM27

Figure 1.4 Arduino IDE software

2. Measurement of voltage

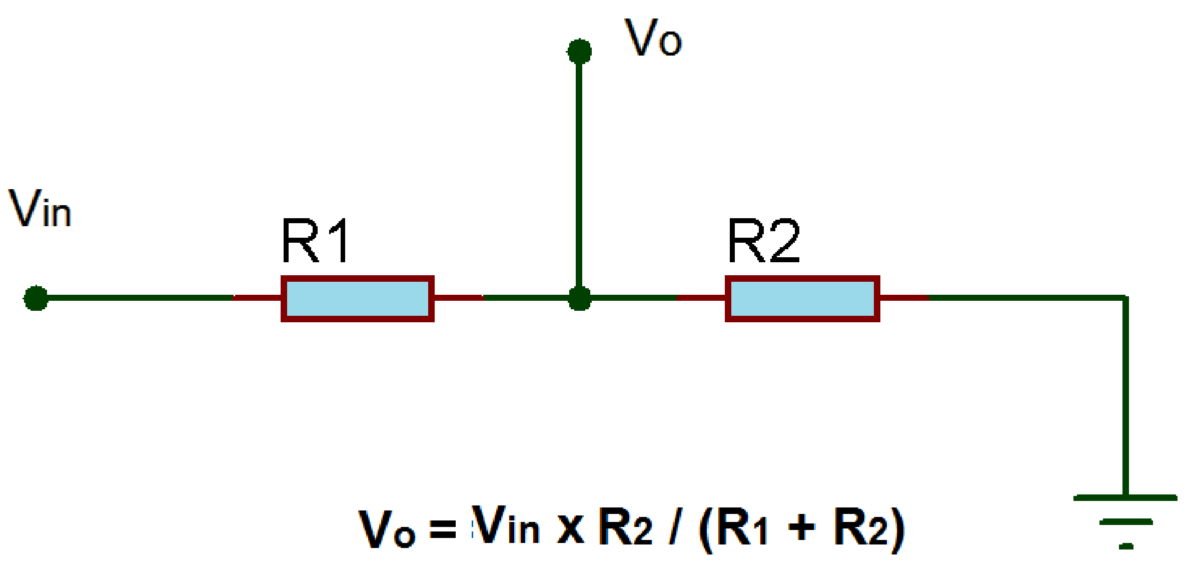
Voltage in power system varies from low voltage 5v to high voltage 400v which varies continuously. The maximum voltage sensed by Arduino is 5v hence we need a method to convert high voltage to low voltage.

2.1 circuit design

Both DC and AC voltages are analogue quantities. Arduino can measure analog DC voltages from 0 to 5 Volts via its analog to digital converter

Voltage divider is the key word here. It can calm down voltage by dividing it in a linear behavior. Voltage divider consists of two resistors (R1 & R2). The output voltage (Vo) can be calculated by the following formula:

$\mathbf{V_0 = V_{in} \times R_2 / (R_1 + R_2)}$



. Figure 2.1 voltage divider circuit

By selecting the appropriate value of resistors we can limit the value of voltage and current. The low value of voltage is given to the analog pin which is connected to the ADC of 10 bit resolution which can divide the 5v in to 1023 parts hence minimum count is of 5/1023 volt.

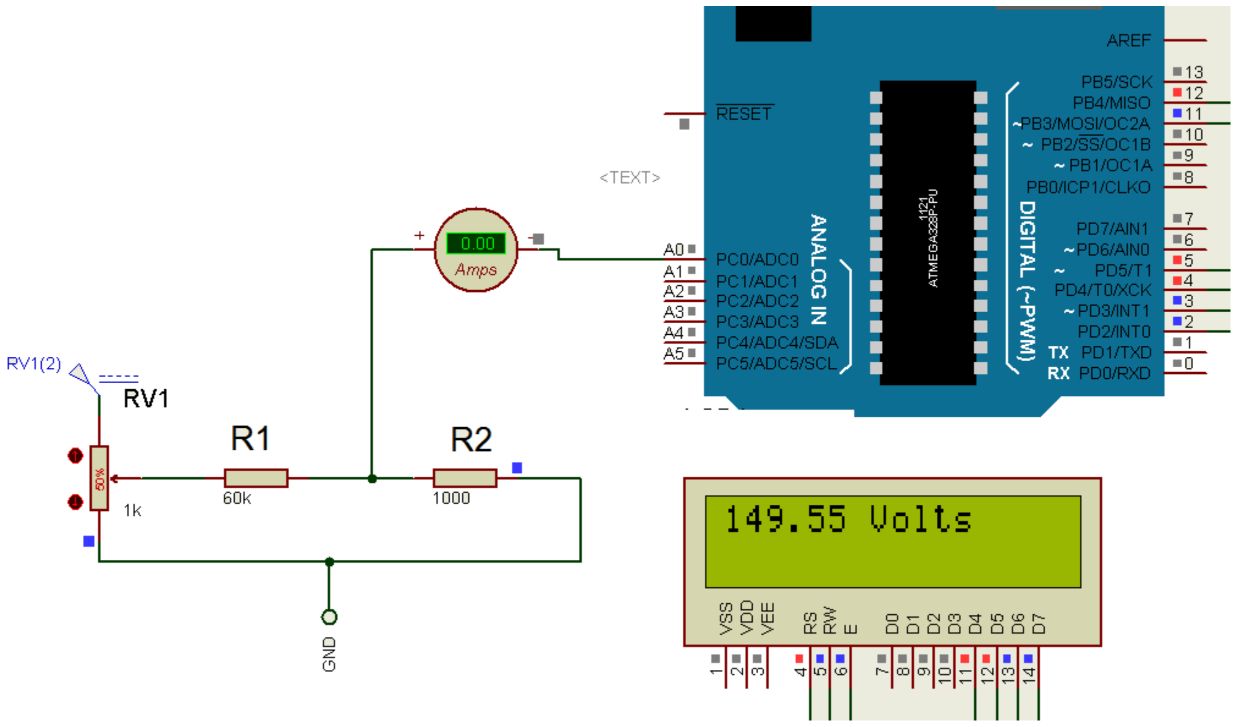


Figure 2.2 Circuit to measure voltage

2.2 Program

Code is written using the c language and library titled Liquid Crystal is used for displaying voltage on LCD screen.

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,5,4,3,2);  // pins for RS, E, DB4, DB5, DB6, DB7

void setup()

{

   lcd.begin(16, 2);

   lcd.clear();

   pinMode(led,OUTPUT);

}

void loop()

{

float sensorValue = analogRead(A0);

float V = sensorValue * (5.0 / 1023.0);

 V=V*100;

lcd.setCursor(1,0);

  lcd.print("voltage");

  lcd.setCursor(10,0);

  lcd.print(V);

  delay(1000);

  lcd.clear();

}
```

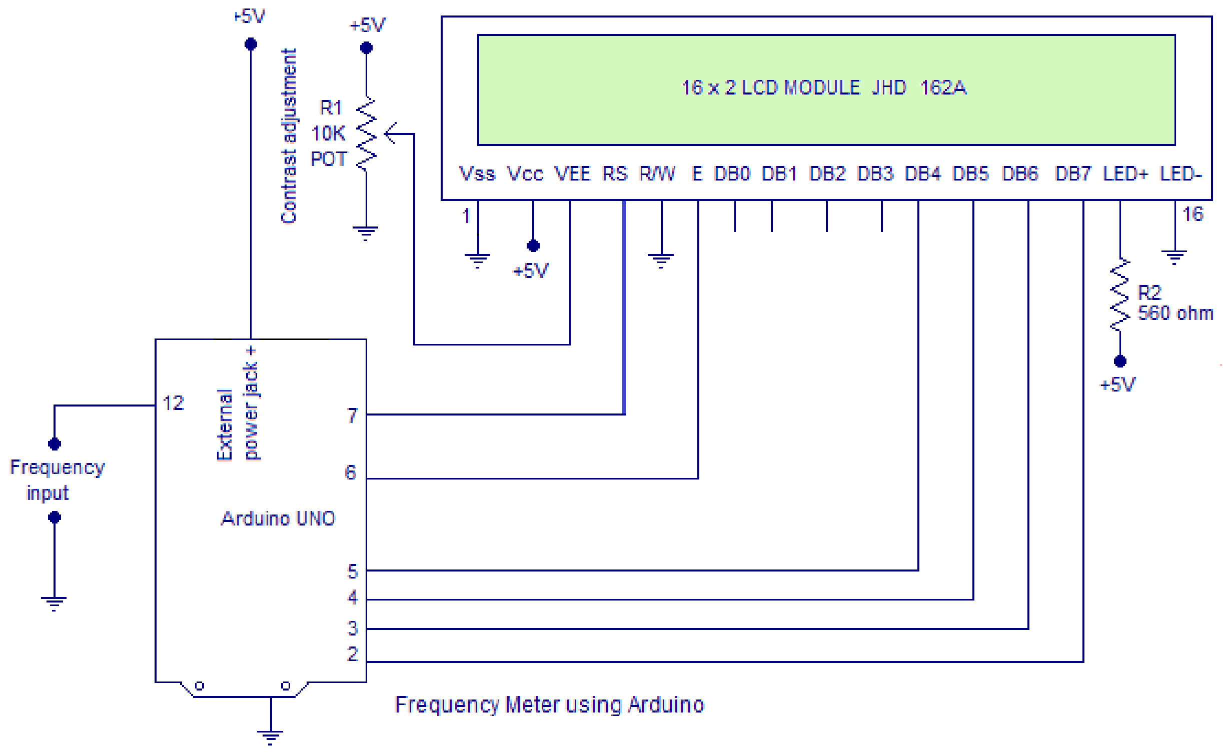
3 Measurement of frequency

When electrical system has AC we need to measure frequency. Every electrical device has a particular operating frequency hence a precise measurement of frequency is needed for the safe operation of electrical system.

Measurement of frequency is done by making the voltage supply low and giving the offset voltage to make the wave in positive side. Arduino can't read the negative wave hence by giving a suitable offset voltage input wave can be made positive.

3.1 Circuit Design

The input frequency wave signal is connected to the digital pin 12 and other terminal is connected to ground



+5V
+5V
Contrast adjustment
R1
10K
POT
16 x 2 LCD MODULE JHD 162A
Vss Vcc VEE RS R/W E DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7 LED+ LED-
1
16
+5V
R2
560 ohm
+5V
External power jack +
12
7
6
5
4
3
2
Frequency input
Arduino UNO
Frequency Meter using Arduino

Figure 3.1 Circuit for frequency measurement

3.2 Program

```
#include <LiquidCrystal.h>

unsigned long input=12;

unsigned long high_time;

unsigned long low_time;

float time_period;

int frequency;

LiquidCrystal lcd(7,6,5,4,3,2);

void setup()

{Serial.begin(9600);

pinMode(input,INPUT);

lcd.begin(16, 2);

}

void loop()

{

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Frequency Meter");

high_time=pulseIn(input,HIGH);

low_time=pulseIn(input,LOW);

time_period=high_time+low_time;

time_period=time_period/1000;

frequency=1000/time_period;

lcd.setCursor(0,1);

lcd.print(frequency);

lcd.print(" Hz");

Serial.print(frequency);

delay(500);}
```

4 Measurement of current

4.1 TALENA AC1020

For the measurement of current here current transformer is used model no Talena AC1020.it have the turn ratio 0f 1000:1 .This is a three terminal device in which one and two are used for the low value voltage measurement. The wire through which current is to be measured is placed inside the hole of CT than output voltage is sensed by the Arduino analog pin .from the voltage current relationship current can be measured.

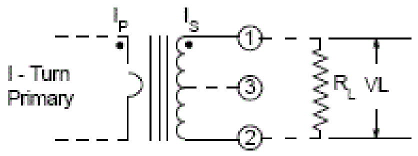


Figure 4.1 Talena AC1020 circuit diagram

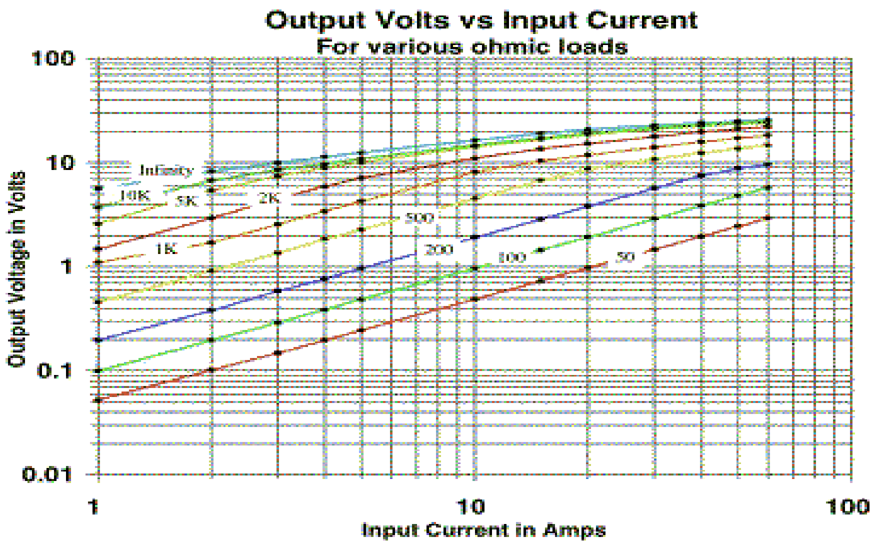


Figure 4.2 output volts vs. input current graph

4.2 AllegroACS710

The Allegro ACS710 current sensor provides economical and precise means for current sensing applications in industrial, commercial, and communications systems. The device is offered in a small footprint surface mount package that allows easy implementation in customer applications.

The ACS710 consists of a precision linear Hall sensor integrated circuit with a copper conduction path located near the surface of the silicon die. Applied current flows through the copper conduction path, and the analog output voltage from the Hall sensor linearly tracks the magnetic field generated by the applied current. The accuracy of the ACS710 is maximized with this patented packaging configuration because the Hall element is situated in extremely close proximity to the current to be measured.

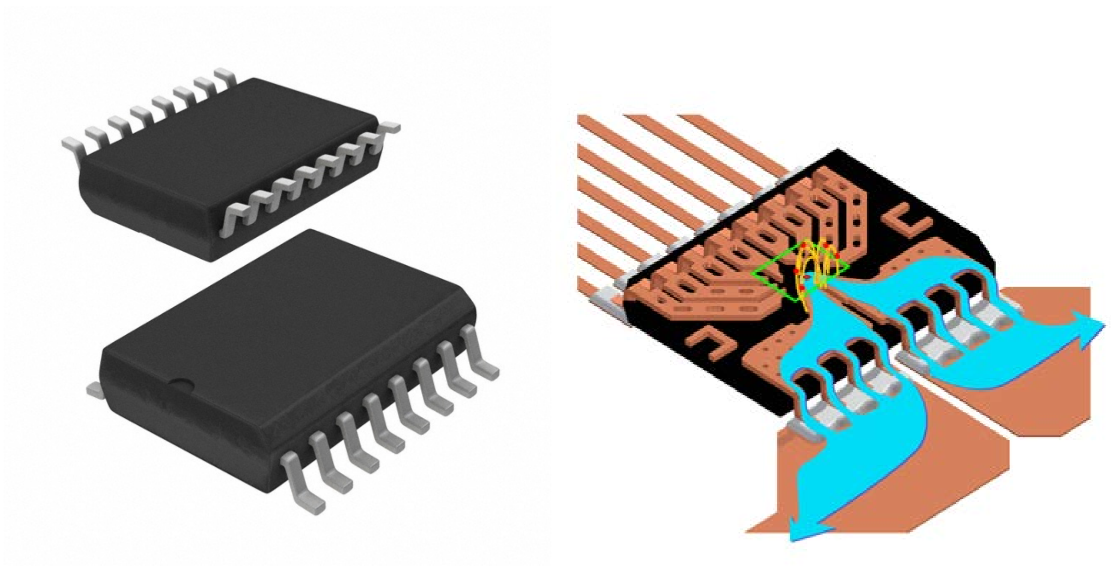


Figure 4.3 ACS710

4.2.1 Measuring Circuit

ACS 710 is a 16 pin IC. Pin 1, 2, 3 and 4 are connected with the input and pin 5,6,7,8 are connected with output of current. To measure current it have VOUT pin which is number 12, it gives the analog voltage which is sensed by the ADC. The supply voltage (VCC) for the operation of IC is 0 to 8 volt DC. Pin 15 is connected with VCC and pin 10 is connected with ground. It can measure current up to 12 A both AC and DC.the output voltage is ratio metric ,it means the voltage is proportional to the supply voltage ,if input voltage is 5 volt than at 5/2=2.5 volt output input current is 0 A . The sensitivity value is 56mV/A at VCC= 5 volt.

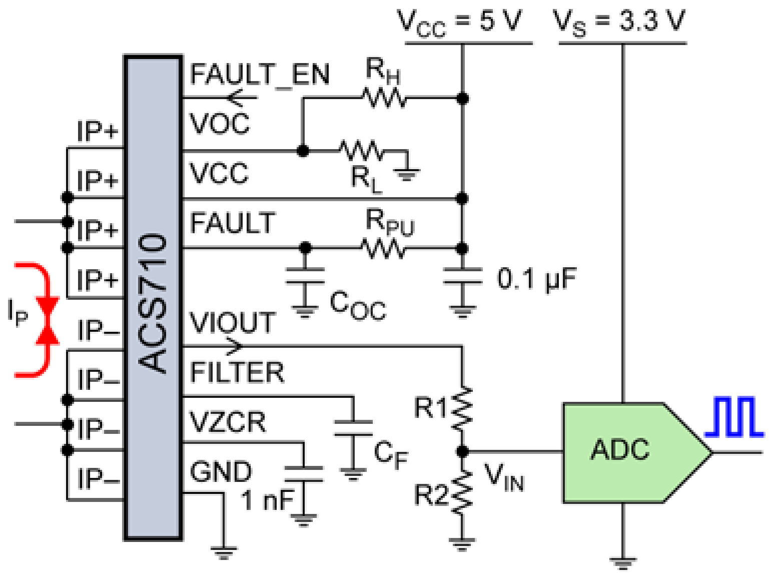


Figure 4.4. Current measurement using ACS 710

5 Measurement of energy

Energy measurement is challenging and necessary parameter for every electrical system. for the measurement of electrical energy digital techniques are used which needs adder, multiplier .value of current and voltage is measured previously which are multiplied to get power, now the value of power is integrated with time to get energy.

E= Pt

P= power measured (KW)

T=time of electricity consumption (h)

E= Electrical energy KWH

5.1 Time

Time is a fundamental parameter which is measured from a long time ago. Here time is measured with the help of RTC (real time clock) module .

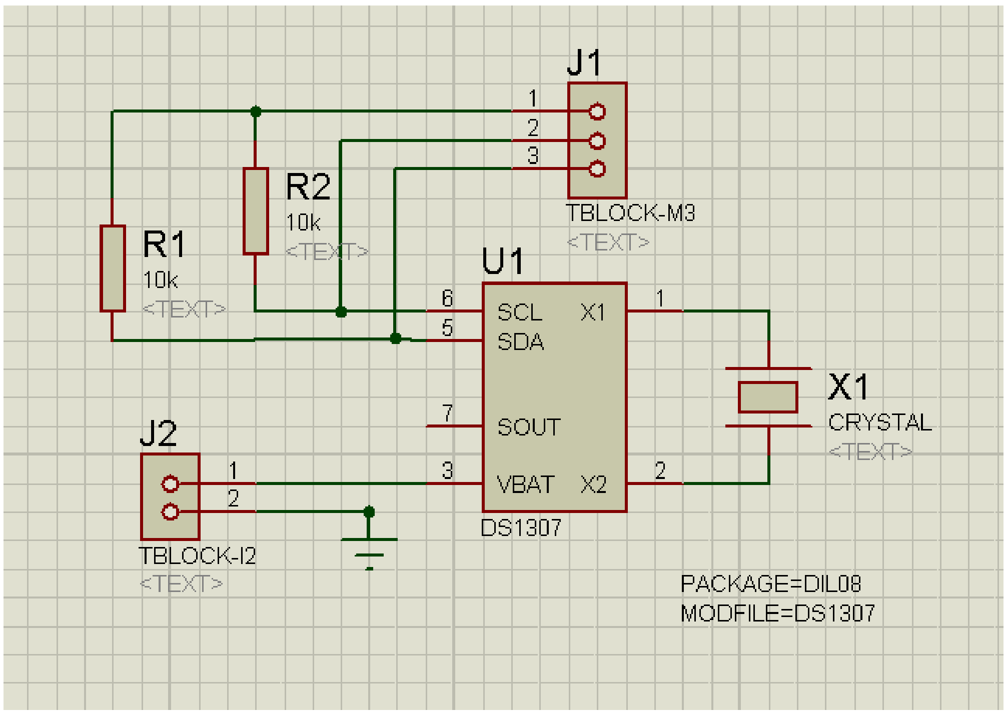


Figure 5.1 Schematic diagram of rtc module

5.1.1 DS1307

The DS1307 serial real-time clock (RTC) is a low power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I2 C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12- hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply.

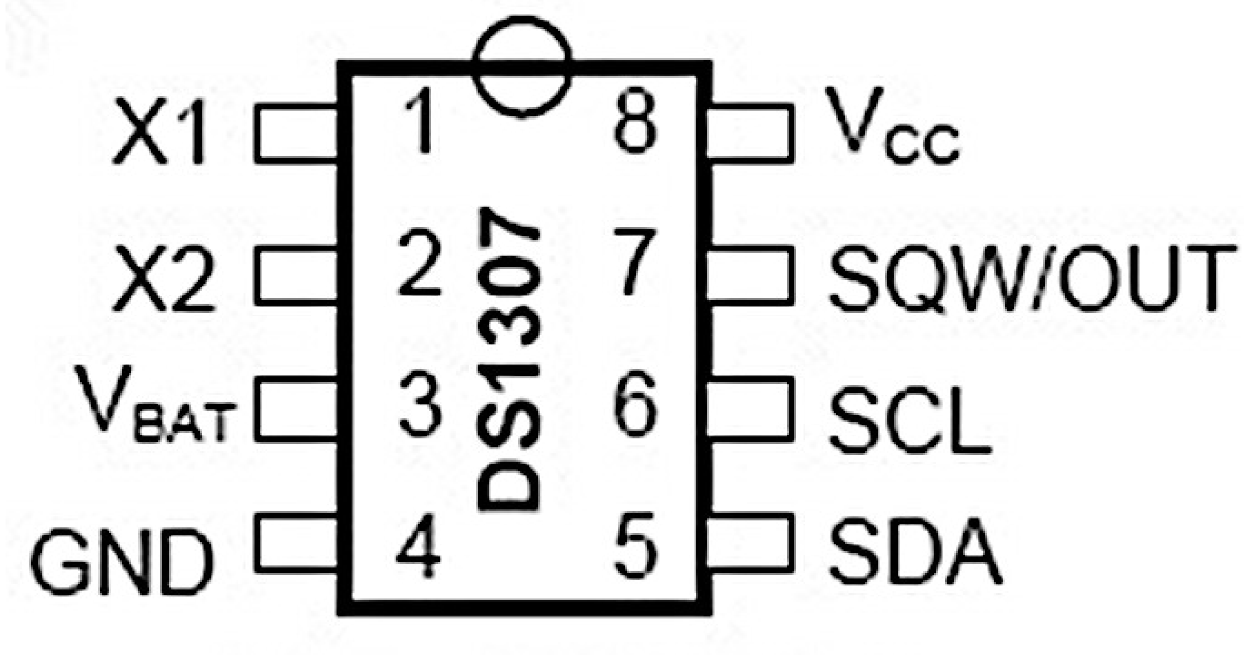


Figure 5.2 Pin configuration for DS1307

X1,X2 -	crystal oscillator	SCL -	serial clock
Vbat-	3 volt button cell	SDA-	serial data
Vcc -	+5 volt supply	GND -	ground
SQW/OUT-	square wave output		

5.1.2 I2C INTERFACE

I2C is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems. It was invented by Philips and now it is used by almost all major IC manufacturers. Each I2C slave device needs an address – they must still be obtained from NXP (formerly Philips semiconductors).

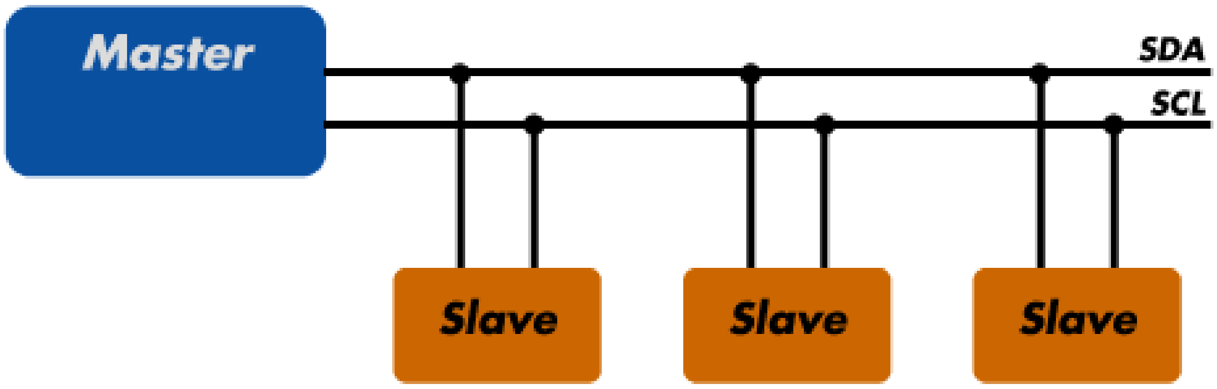


Figure 5.3 I2C BUS

5.1.3 RTC WITH ARDUINO

Arduino have two pins allocated for I2C interface which are A4 and A5. A4 is connected with serial data and A5 is connected with serial clock .VCC can be given from the +5 volt arduino pin. All ground are common at one point.3 volt button cell have a life of about 10 years hence when the main supply is switched off rtc can run for the long time using power from the cell. after making the hardware connection we need to upload the program to the arduino .The library used here is “RealTimeClockDS1307”,which have all the code needed for rtc.

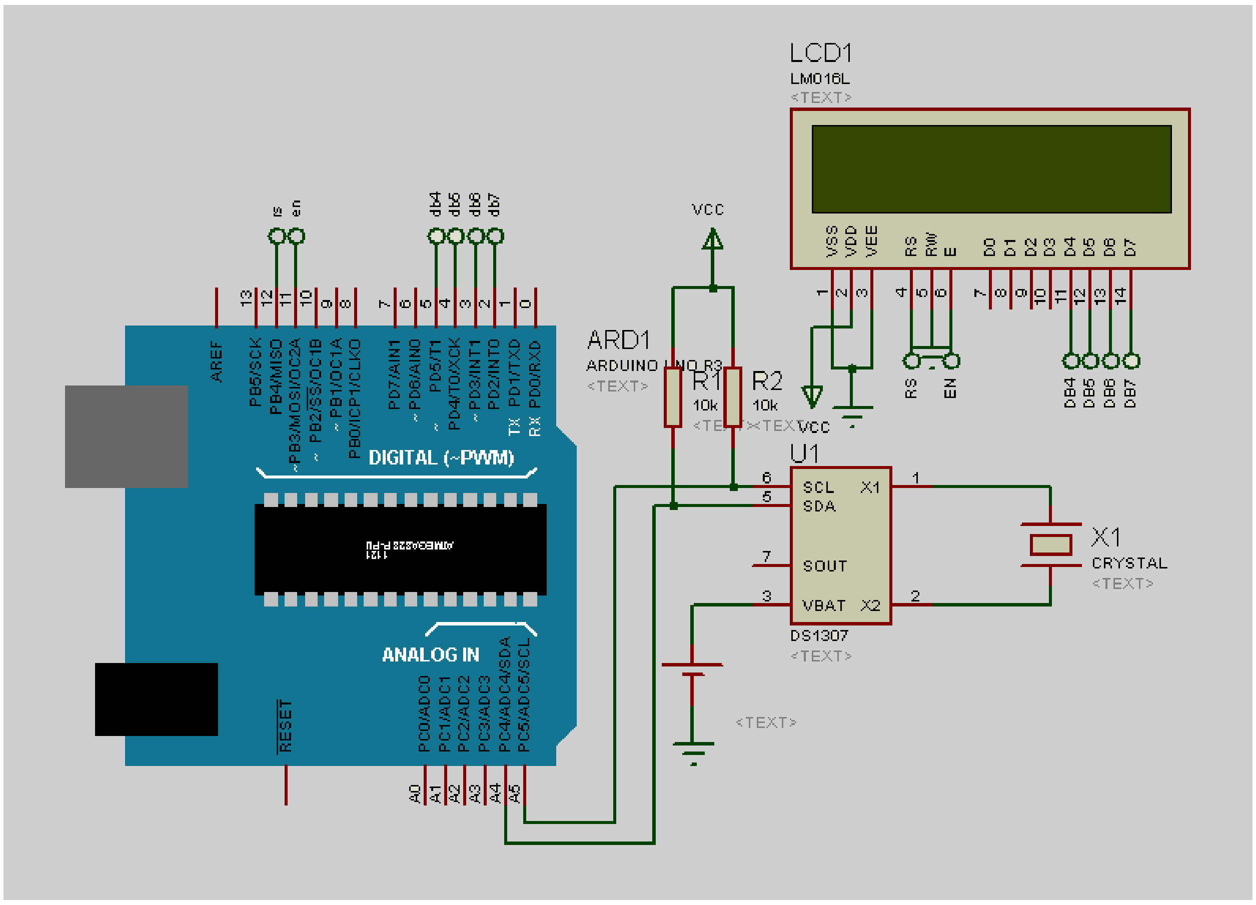


Figure 5.4 Connecting rtc with arduino

5.1.4 program

```
#include <Wire.h>
#include <RealTimeClockDS1307.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(4,5,6,7,8,9);

//RealTimeClock RTC;//=new RealTimeClock();
#define Display_Clock_Every_N_Seconds 1
#define Display_ShortHelp_Every_N_Seconds 60

int count=0;
char formatted[] = "00-00-00 00:00:00x";

void setup() {
//  Wire.begin();
```

```
  Serial.begin(9600);
  lcd.begin(16, 2);
}

void loop() {
  if(Serial.available())
  {
    processCommand();
  }
  delay(1000);
  RTC.readClock();
  count++;
  if(count % Display_Clock_Every_N_Seconds == 0){
    Serial.print(count);
    Serial.print(": ");
    RTC.getFormatted(formatted);
    Serial.print(formatted);
    Serial.println();
    //lcd.setCursor(0,0);
    //lcd.print("DATE(YY/M/D) TIME");
    lcd.setCursor(0,0);
    lcd.print(formatted);

  }

  if(count % Display_ShortHelp_Every_N_Seconds == 0) {
    Serial.println("Send ? for a list of commands.");
  }
}

void processCommand() {
  if(!Serial.available()) { return; }
  char command = Serial.read();
  int in,in2;
```

```
switch(command)
{
  case 'H':
  case 'h':
  in=SerialReadPosInt();
  RTC.setHours(in);
  RTC.setClock();
  Serial.print("Setting hours to ");
  Serial.println(in);
  break;
  case 'I':
  case 'i':
  in=SerialReadPosInt();
  RTC.setMinutes(in);
  RTC.setClock();
  Serial.print("Setting minutes to ");
  Serial.println(in);
  break;
  case 'S':
  case 's':
  in=SerialReadPosInt();
  RTC.setSeconds(in);
  RTC.setClock();
  Serial.print("Setting seconds to ");
  Serial.println(in);
  break;
  case 'Y':
  case 'y':
  in=SerialReadPosInt();
  RTC.setYear(in);
  RTC.setClock();
  Serial.print("Setting year to ");
  Serial.println(in);
  break;
```

```
case 'M':
case 'm':
in=SerialReadPosInt();
RTC.setMonth(in);
RTC.setClock();
Serial.print("Setting month to ");
Serial.println(in);
break;
case 'D':
case 'd':
in=SerialReadPosInt();
RTC.setDate(in);
RTC.setClock();
Serial.print("Setting date to ");
Serial.println(in);
break;
case 'W':
Serial.print("Day of week is ");
Serial.println((int) RTC.getDayOfWeek());
break;
case 'w':
in=SerialReadPosInt();
RTC.setDayOfWeek(in);
RTC.setClock();
Serial.print("Setting day of week to ");
Serial.println(in);
break;

case 't':
case 'T':
if(RTC.is12hour()) {
  RTC.switchTo24h();
  Serial.println("Switching to 24-hour clock.");
} else {
  RTC.switchTo12h();
  Serial.println("Switching to 12-hour clock.");
}
RTC.setClock();
break;

case 'A':
case 'a':
if(RTC.is12hour()) {
  RTC.setAM();
  RTC.setClock();
  Serial.println("Set AM.");
} else {
  Serial.println("(Set hours only in 24-hour mode.)");
}
break;

case 'P':
case 'p':
if(RTC.is12hour()) {
  RTC.setPM();
  RTC.setClock();
  Serial.println("Set PM.");
} else {
  Serial.println("(Set hours only in 24-hour mode.)");
}
break;

case 'q':
RTC.sqwEnable(RTC.SQW_1Hz);
Serial.println("Square wave output set to 1Hz");
break;
case 'Q':
RTC.sqwDisable(0);
```

```
Serial.println("Square wave output disabled (low)");
break;

case 'z':
RTC.start();
Serial.println("Clock oscillator started.");
break;
case 'Z':
RTC.stop();
Serial.println("Clock oscillator stopped.");
break;

case '>':
in=SerialReadPosInt();
in2=SerialReadPosInt();
RTC.writeData(in, in2);
Serial.print("Write to register ");
Serial.print(in);
Serial.print(" the value ");
Serial.println(in2);
break;
case '<':
in=SerialReadPosInt();
in2=RTC.readData(in);
Serial.print("Read from register ");
Serial.print(in);
Serial.print(" the value ");
Serial.println(in2);
break;

default:
Serial.println("Unknown command. Try these:");
Serial.println(" h## - set Hours       d## - set Date");
Serial.println(" i## - set mInutes     m## - set Month");
```

```
    Serial.println(" s## - set Seconds     y## - set Year");
    Serial.println(" w## - set arbitrary day of Week");
    Serial.println(" t   - toggle 24-hour mode");
    Serial.println(" a   - set AM          p   - set PM");
    Serial.println();
    Serial.println(" z   - start clock     Z   - stop clock");
    Serial.println(" q   - SQW/OUT = 1Hz   Q   - stop SQW/OUT");
    Serial.println();
    Serial.println(" >##,###  - write to register ## the value ###");
    Serial.println(" <##      - read the value in register ##");

  }//switch on command

}

//read in numeric characters until something else
//or no more data is available on serial.
int SerialReadPosInt() {
  int i = 0;
  boolean done=false;
  while(Serial.available() && !done)
  {
    char c = Serial.read();
    if (c >= '0' && c <='9')
    {
      i = i * 10 + (c-'0');
    }
    else
    {
      done = true;
    }
  }
  return i;
}
```

5.2 Arduino memory

There are three pools of memory in the microcontroller used on avr-based Arduino boards

- Flash memory (program space), is where the Arduino sketch is stored.
- SRAM (static random access memory) is where the sketch creates and manipulates variables when it runs.
- EEPROM is memory space that programmers can use to store long-term information.

Flash memory and EEPROM memory are non-volatile (the information persists after the power is turned off). SRAM is volatile and will be lost when the power is cycled. The ATmega328 chip found on the Uno has the following amounts of memory:

Flash 32k bytes (of which .5k is used for the boot loader)

SRAM   2k bytes

EEPROM 1k byte

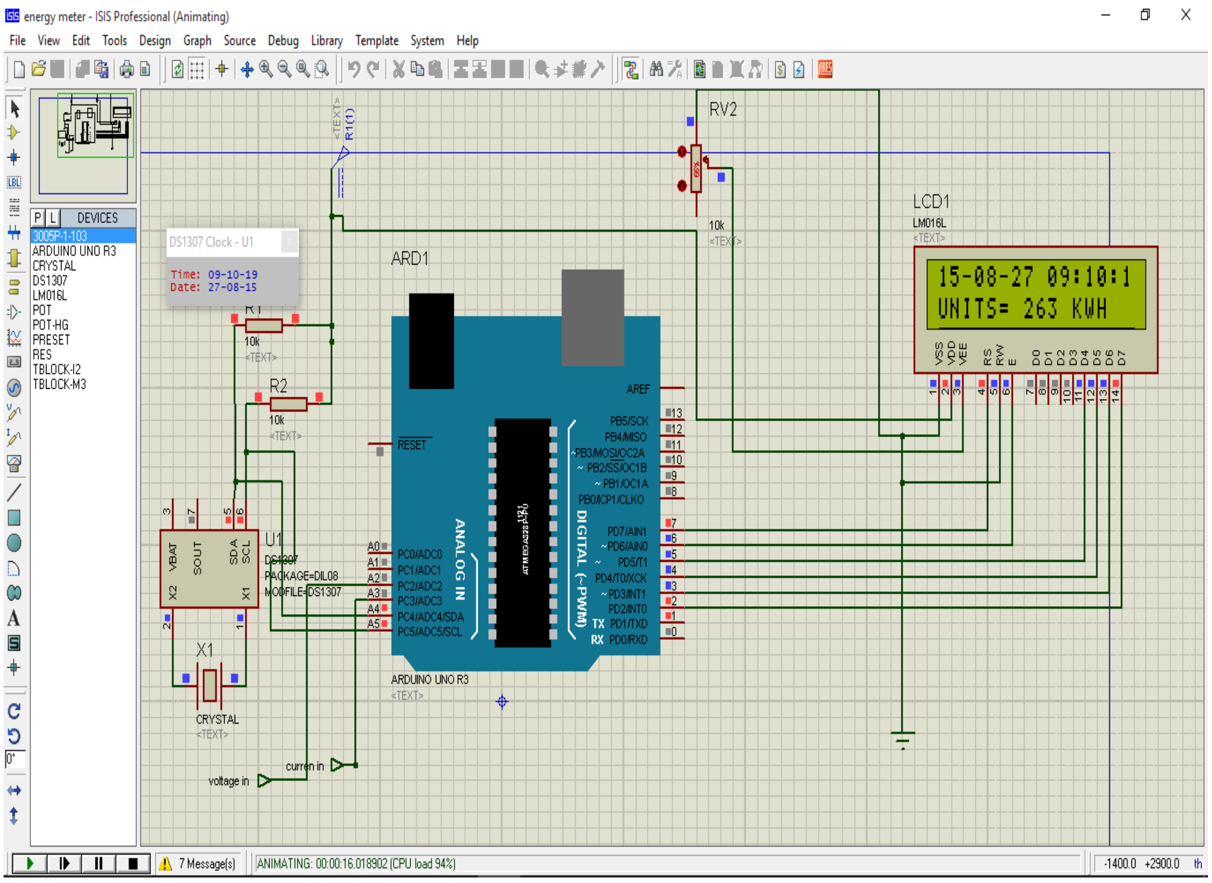
For the reliable measuremet of energy we need to restore the value of energy measured last time when power is cut off, for that arduino have 1K EEPROM which can be used to store the value of energy in real time. Arduino have a library named ”EEPROM.h” used to read and write the data in memory .

5.2.1 program

```
/*
 * EEPROM Write
 *
 * Stores value of of a constant integer
 * This value will stay in the EEPROM when the board is
 * turned off and may be retrieved later by another sketch.
 */
#include <EEPROM.h>

/** the current address in the EEPROM (i.e. which byte we're going to write to next) **/
int addr = 0;

void setup(){ /** Empty setup. **/}

void loop()
{
  int val = 100;

  EEPROM.write(addr, val);
   val = EEPROM.read(addr);
delay(100);
}
```

5.3 ENERGY CIRCUIT

The circuit is designed using proteus software 7. The complete setup include arduino uno microcontroller,rtc module,lcd panel,input voltage and current and 5 volt dc supply to power modules. All the modules are connected as shown in circuit diagram.



energy meter - ISIS Professional (Animating)

DS1307 Clock - U1

```
Time: 09-10-19
Date: 27-08-15
```

LCD1 LM016L:

```
15-08-27 09:10:1
UNITS= 263 KWH
```

7 Message(s)

ANIMATING: 00:00:16.018902 (CPU load 94%)

Figure 5.5 Simulation result of energy module

5.3.1 program

```
#include <EEPROM.h>
#include <Wire.h>
#include <RealTimeClockDS1307.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(7,6,5,4,3,2);

//RealTimeClock RTC;//=new RealTimeClock();
#define Display_Clock_Every_N_Seconds 1
#define Display_ShortHelp_Every_N_Seconds 60

int count=0;
char formatted[] = "00-00-00 00:00:00x";
unsigned int  energy=0,energy1=0 ;
int addr = 0;



void setup() {
//  Wire.begin();
  Serial.begin(9600);
  lcd.begin(16, 2);
}

void loop() {
  if(Serial.available())
  {
    processCommand();
  }
  delay(1000);
  RTC.readClock();
  count++;
  if(count % Display_Clock_Every_N_Seconds == 0){
    Serial.print(count);
    Serial.print(": ");
    RTC.getFormatted(formatted);
    Serial.print(formatted);
    Serial.println();
    //lcd.setCursor(0,0);
    //lcd.print("DATE(YY/M/D) TIME");
    lcd.setCursor(0,0);
    lcd.print(formatted);

    if(count==1)
    {
      energy1=EEPROM.read(addr);
   energy=energy1+ (2000.00*count)/3600.00;
    }
    else
    {

      energy=energy1+(2000.00*count)/3600.00;
    }
    EEPROM.write(addr, energy);
    if(energy>=255)
    {addr=addr+1;
      }

    lcd.setCursor(0,1);
    lcd.print("UNITS= ");
    lcd.print(energy);
    lcd.print(" KWH");

    Serial.print(energy);
    Serial.println();

  }
```

```
  if(count % Display_ShortHelp_Every_N_Seconds == 0) {
    Serial.println("Send ? for a list of commands.");
  }
}

void processCommand() {
  if(!Serial.available()) { return; }
  char command = Serial.read();
  int in,in2;
  switch(command)
  {
    case 'H':
    case 'h':
    in=SerialReadPosInt();
    RTC.setHours(in);
    RTC.setClock();
    Serial.print("Setting hours to ");
    Serial.println(in);
    break;
    case 'I':
    case 'i':
    in=SerialReadPosInt();
    RTC.setMinutes(in);
    RTC.setClock();
    Serial.print("Setting minutes to ");
    Serial.println(in);
    break;
    case 'S':
    case 's':
    in=SerialReadPosInt();
    RTC.setSeconds(in);
    RTC.setClock();
    Serial.print("Setting seconds to ");
    Serial.println(in);
```

```
break;
case 'Y':
case 'y':
in=SerialReadPosInt();
RTC.setYear(in);
RTC.setClock();
Serial.print("Setting year to ");
Serial.println(in);
break;
case 'M':
case 'm':
in=SerialReadPosInt();
RTC.setMonth(in);
RTC.setClock();
Serial.print("Setting month to ");
Serial.println(in);
break;
case 'D':
case 'd':
in=SerialReadPosInt();
RTC.setDate(in);
RTC.setClock();
Serial.print("Setting date to ");
Serial.println(in);
break;
case 'W':
Serial.print("Day of week is ");
Serial.println((int) RTC.getDayOfWeek());
break;
case 'w':
in=SerialReadPosInt();
RTC.setDayOfWeek(in);
RTC.setClock();
Serial.print("Setting day of week to ");
Serial.println(in);
break;

case 't':
case 'T':
if(RTC.is12hour()) {
  RTC.switchTo24h();
  Serial.println("Switching to 24-hour clock.");
} else {
  RTC.switchTo12h();
  Serial.println("Switching to 12-hour clock.");
}
RTC.setClock();
break;

case 'A':
case 'a':
if(RTC.is12hour()) {
  RTC.setAM();
  RTC.setClock();
  Serial.println("Set AM.");
} else {
  Serial.println("(Set hours only in 24-hour mode.)");
}
break;

case 'P':
case 'p':
if(RTC.is12hour()) {
  RTC.setPM();
  RTC.setClock();
  Serial.println("Set PM.");
} else {
  Serial.println("(Set hours only in 24-hour mode.)");
}
break;

case 'q':
RTC.sqwEnable(RTC.SQW_1Hz);
Serial.println("Square wave output set to 1Hz");
break;
case 'Q':
RTC.sqwDisable(0);
Serial.println("Square wave output disabled (low)");
break;

case 'z':
RTC.start();
Serial.println("Clock oscillator started.");
break;
case 'Z':
RTC.stop();
Serial.println("Clock oscillator stopped.");
break;

case '>':
in=SerialReadPosInt();
in2=SerialReadPosInt();
RTC.writeData(in, in2);
Serial.print("Write to register ");
Serial.print(in);
Serial.print(" the value ");
Serial.println(in2);
break;
case '<':
in=SerialReadPosInt();
in2=RTC.readData(in);
Serial.print("Read from register ");
```

```
    Serial.print(in);
    Serial.print(" the value ");
    Serial.println(in2);
    break;

    default:
    Serial.println("Unknown command. Try these:");
    Serial.println(" h## - set Hours       d## - set Date");
    Serial.println(" i## - set mInutes     m## - set Month");
    Serial.println(" s## - set Seconds     y## - set Year");
    Serial.println(" w## - set arbitrary day of Week");
    Serial.println(" t   - toggle 24-hour mode");
    Serial.println(" a   - set AM          p   - set PM");
    Serial.println();
    Serial.println(" z   - start clock     Z   - stop clock");
    Serial.println(" q   - SQW/OUT = 1Hz   Q   - stop SQW/OUT");
    Serial.println();
    Serial.println(" >##,###  - write to register ## the value ###");
    Serial.println(" <##      - read the value in register ##");

  }//switch on command

}

//read in numeric characters until something else
//or no more data is available on serial.
int SerialReadPosInt() {
  int i = 0;
  boolean done=false;
  while(Serial.available() && !done)
  {
    char c = Serial.read();
    if (c >= '0' && c <='9')
    {
      i = i * 10 + (c-'0');
    }
    else
    {
      done = true;
    }
  }
  return i;
}
```

6. RESULT AND CONCLUSION

- Energy is measured and data is stored using the digital techniques.
- Measurement using digital technique consumes less power and instruments take less space.
- Using microcontroller we can control the devices on the bases of electrical parameter.
- Making the common computer program for the whole system make the measurement easy and reliable.
- We can easily transfer the data from one computer to other.
- Real time data can be read which can be used for better control of power system

7. References

- https://www.arduino.cc
- https://www.schukat.com
- The art of electronics third edition by Paul Horowitz , Winfield Hill