Government Engineering College

Thrissur

GSM ENABLED SMART ENERGY METER

Design Project Report 2017

Guided by Prof. Jose Sebastian

Group Members:

Alin Anto

David Thomas

Arya Narayanan

Eby Jacob

ACKNOWLEDGEMENT

We express our sincere gratitude to our project guide **Prof. Jose Sebastian**, Department of Electrical Engineering, Government Engineering

College, Thrissur for his timely guidance, inspiration and constant support during the course of the project work.

We are also thankful to our group tutor **MrsJayasoorya**, Assistant Professor, Department of Electrical Engineering, Government Engineering College, Thrissur for her continuous encouragement and motivation during the course of the work.

We also wish to express our sincere thanks to **Dr. M. Nandakumar** , Professor and Head, Department of Electrical Engineering, Government Engineering College, Thrissur for providing us with all the necessary facilities to carry out this project.

Our sincere thanks are also due to all faculty members, laboratory staff and our classmates who have been there with us every time, rendering the required help and assistance.

ABSTRACT

The full utility of a household or industrial energy meter can be utilised if it can be interface with a communication device. Since GSM communication is the most portable device based communication, the same can be applied for energy meter. Digital type device is selected to make the meter smart and reliable. Automation has become a necessity in the modern world. Human labour is considered unwanted even in the most simplest or crucial of the tasks which can be automated. This reduces mistakes due to human error. Energy conservation is also a main driving factor which has

led to the selection of this particular product.

CONTENTS

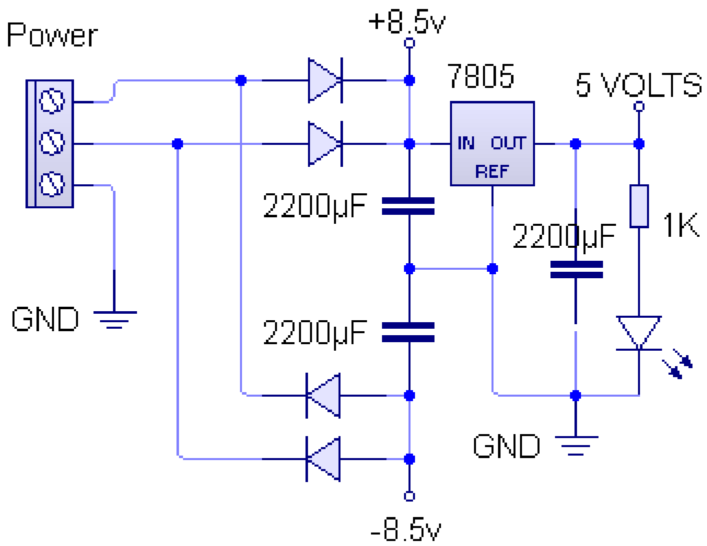
- Stages of design Implementation
    - Design sketching
    - PCB routing
    - PCB etching, soldering, hardware finalisation
    - Software design
    - Testing
- Source Code
- Result
- Conclusion
- References

DESIGN OBJECTIVE

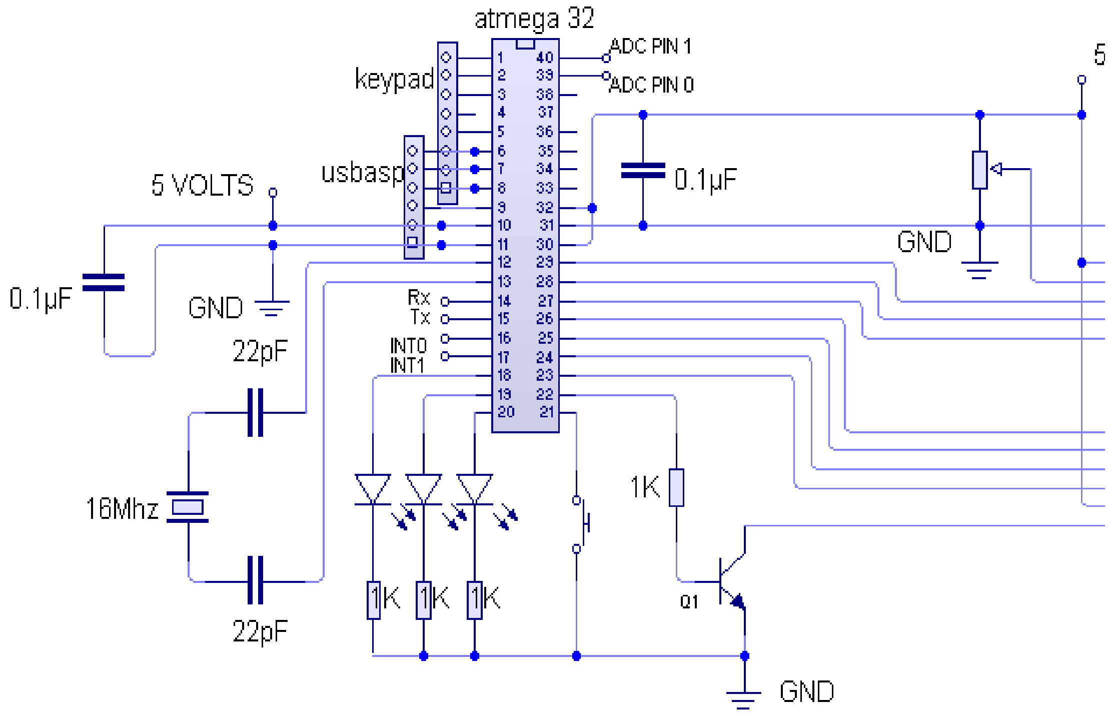
- To make a smart energy meter.
- To implement tariff based calculations.
- To interface energy meter with GSM module.
- To implement communication with user during:
- Low power factor
- Over current
- Billing information

BLOCK DIAGRAM

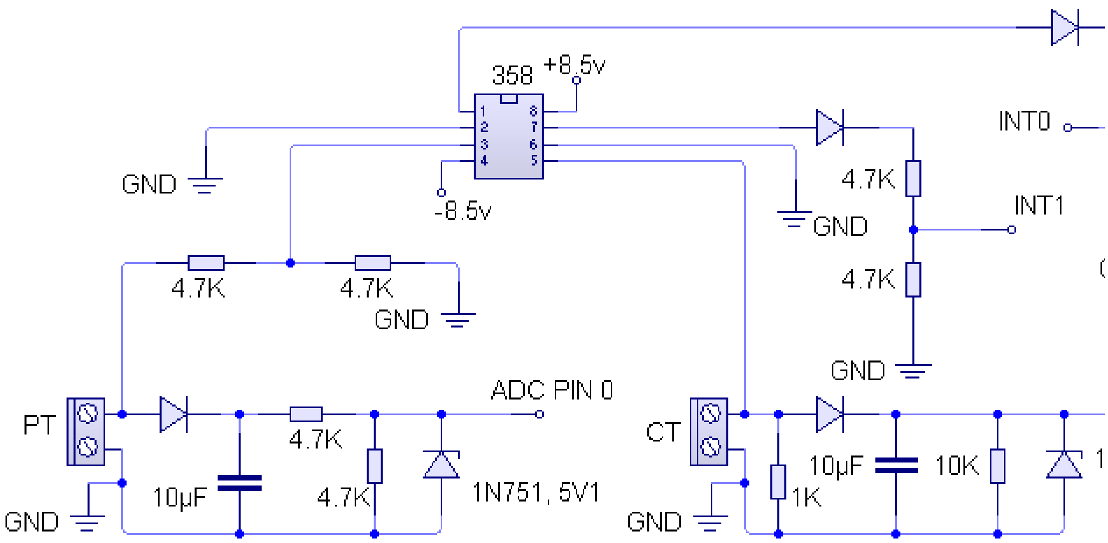
CIRCUIT DIAGRAM



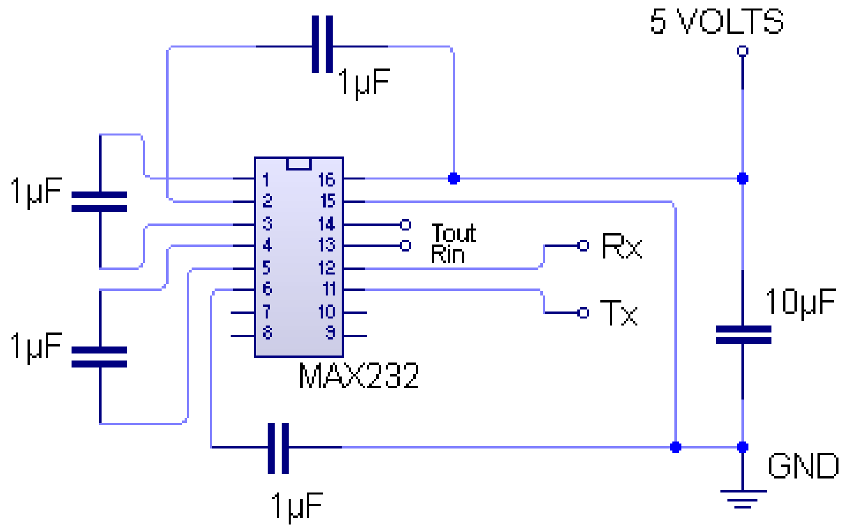
POWER SUPPLY CIRCUIT



MICROCONTROLLER CIRCUIT



CT & PT CIRCUIT



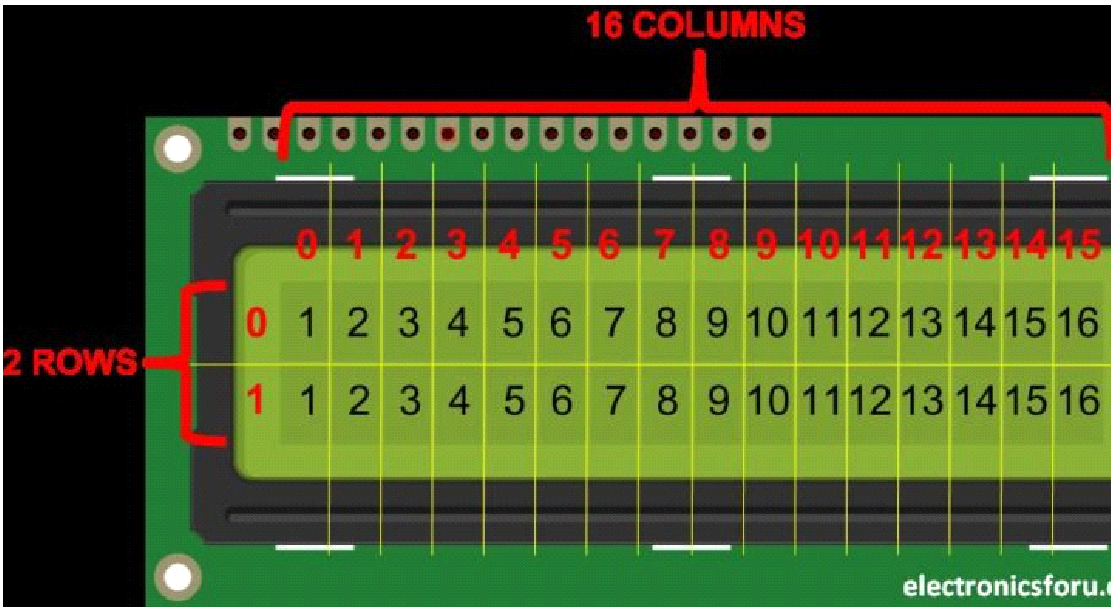
UART INTERFACE CIRCUIT

COMPONENTS

- ATMEGA32
- LCD MODULE
- POTENTIAL TRANSFORMER
- CURRENT TRANSFORMER
- LM358
- RESISTORS,DIODES,CAPACITORS
- ZENER DIODES 1N751
- MAX232
- 7805 VOLTAGE REGULATOR

LCD MODULE

We come across LCD displays everywhere around us. Computers, calculators, television sets, mobile phones, digital watches use some kind of display to display the time. An LCD is an electronic display module which uses liquid crystal to produce a visible image. The 16×2 LCD display is a very basic module commonly used in DIYs and circuits. The 16×2 lcd module can display 16 characters per line in 2 such lines. In this LCD each character is displayed in a 5×7 pixel matrix.



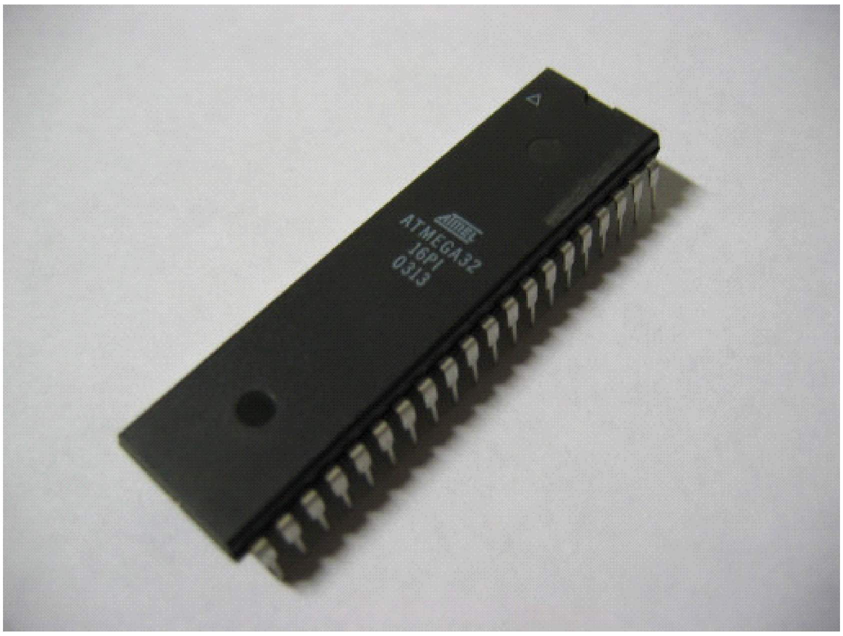
These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

ATMEGA32 MICROCONTROLLER

ATmega32 is very much similar to ATmega16 microcontroller with certain differences which are discussed below. ATmega32 is an 8-bit high performance microcontroller of Atmel's Mega AVR family. Atmega32 is based on enhanced RISC (Reduced Instruction Set Computing) architecture with 131 powerful instructions. Most of the instructions execute in one machine cycle. Atmega32 can work on a maximum frequency of 16MHz.

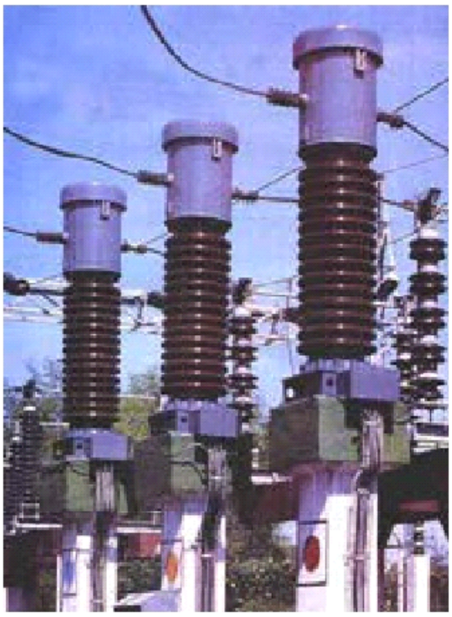
The differences between ATmega32 and ATmega16 can be summarized as follows:

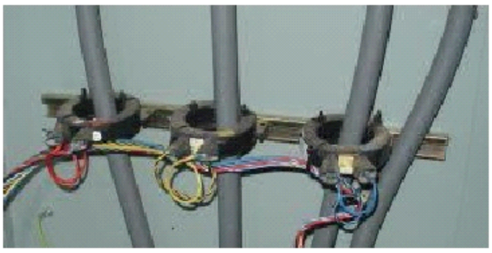
1. ATmega32 has 32 KB programmable flash memory, static RAM of 2 KB and EEPROM of 1 KB. The endurance cycle of flash memory and EEPROM is 10,000 and 100,000, respectively

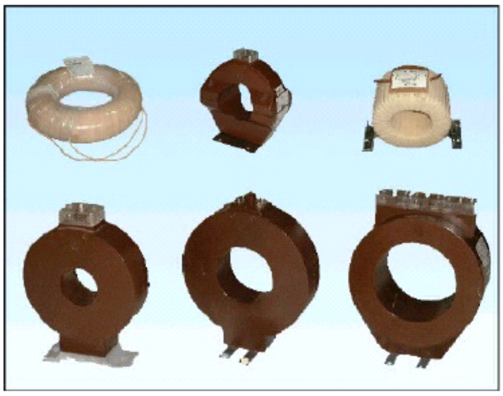


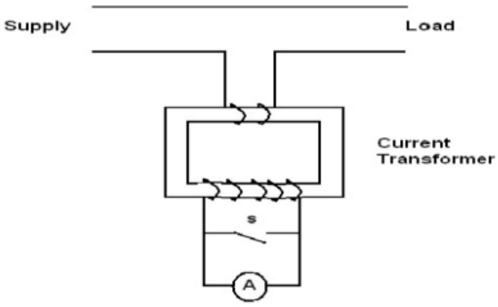
CT AND PT

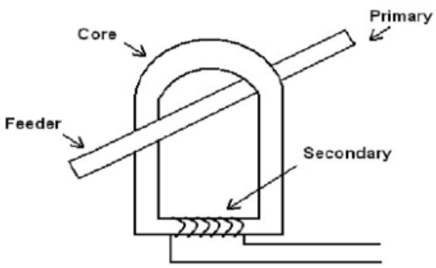
CT isused to measure or monitor the current intransmission lines and to isolate the metering equipmentand relay connected to secondary side











POTENTIAL TRANSFORMER

Voltage **transformers** (VT), also called **potentialtransformers** (PT), are a parallel connected type of instrument **transformer**. They are designed to present negligible load to the supply being measured and have an accurate voltage ratio and phase relationship to enable accurate secondary connected metering.

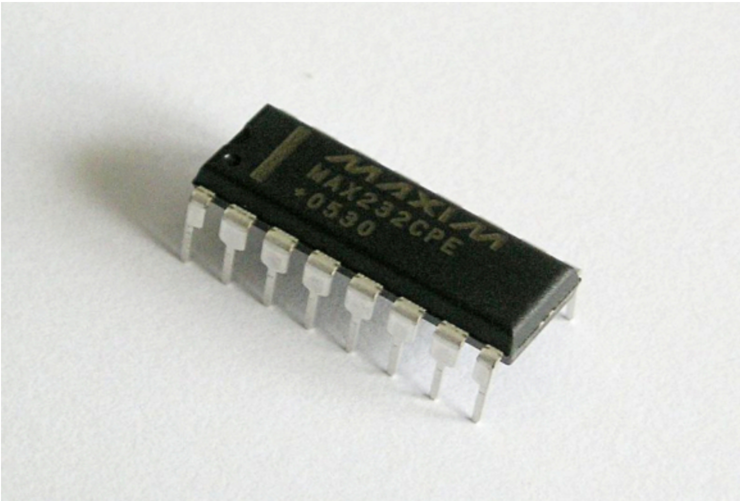
MAX232 IC

The MAX232 is an integrated circuit first created in 1987 by Maxim Integrated Products that converts signals from a TIA-232 (RS-232) serial port to signals suitable for use in TTL-compatible digital logic circuits. The MAX232 is a dual transmitter / dual receiver that typically is used to convert the RX, TX, CTS, RTS signals.

The drivers provide TIA -232 voltage level outputs (about ± 7.5 volts) from a single 5 -volt supply by on-chip charge pumps and external capacitors. This makes it useful for implementing TIA-232 in devices that otherwise do not need any other voltages.

The receivers reduce TIA-232 inputs, which may be as high as ±25 volts, to standard 5 volt TTL levels. These receivers have a typical threshold of 1.3 volts and a typical hysteresis of 0.5 volts.

The MAX232 replaced an older pair of chips MC1488 and MC1489 that performed similar RS-232 translation. The MC1488 quad transmitter chip required 12 volt and -12 volt power,[1] and MC1489 quad receiver chip required 5 volt power.[2] The main disadvantages of this older solution was the +/ - 12 volt power requirement, only supported 5 volt digital logic, and two chips instead of one.

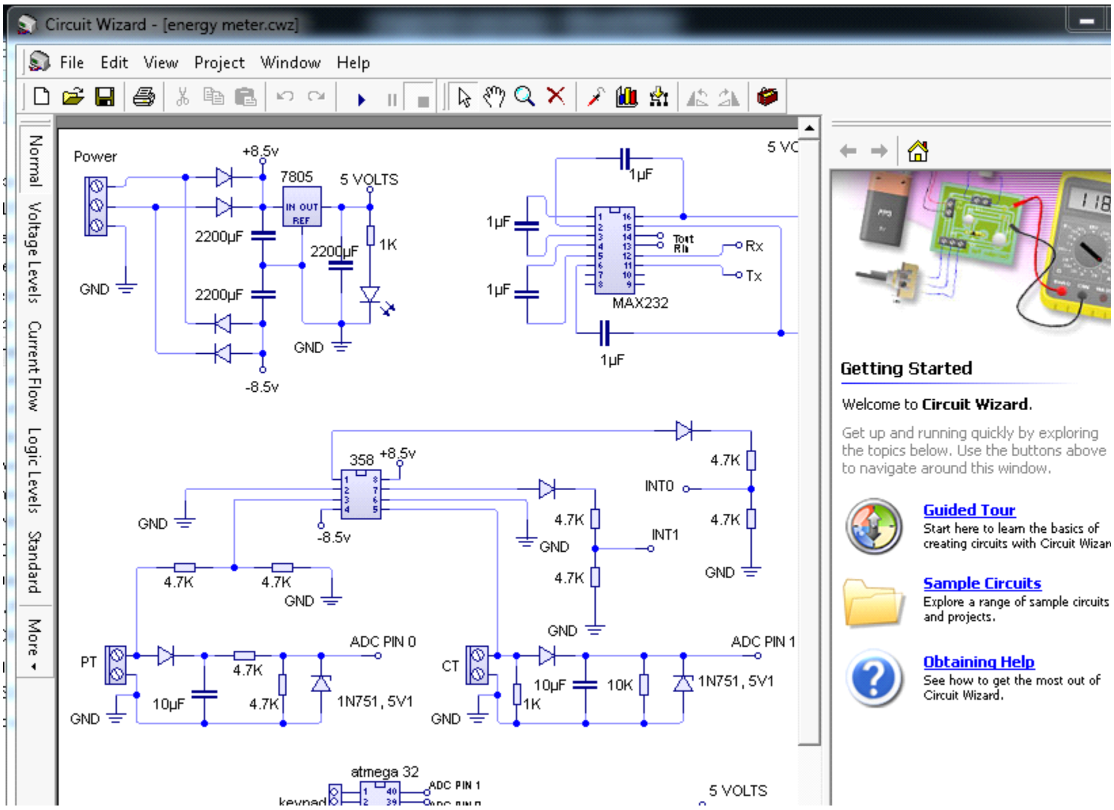


STAGES OF DESIGN IMPLEMENTATION

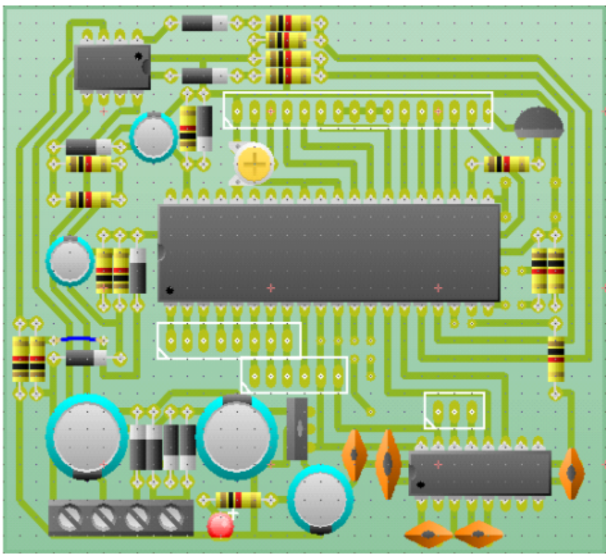
Design sketching & PCB routing

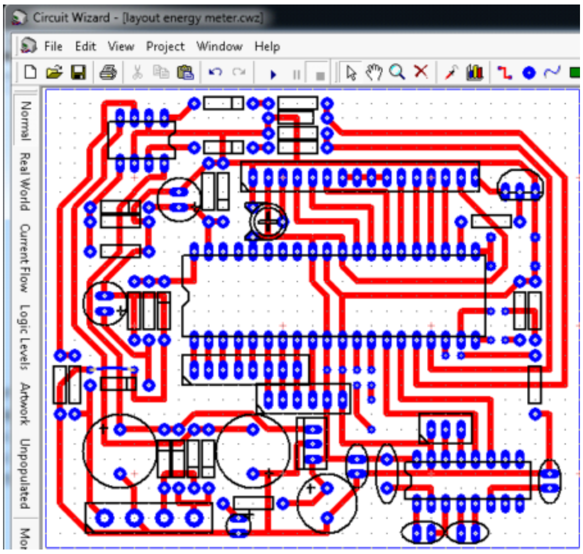
Sketching of the design was done with the help of software called circuit wizard. Circuit Wizard, New wave concept product, is a revolutionary new system that combines circuit design, PCB design, simulation, CAD/CAM manufacture in one complete package.

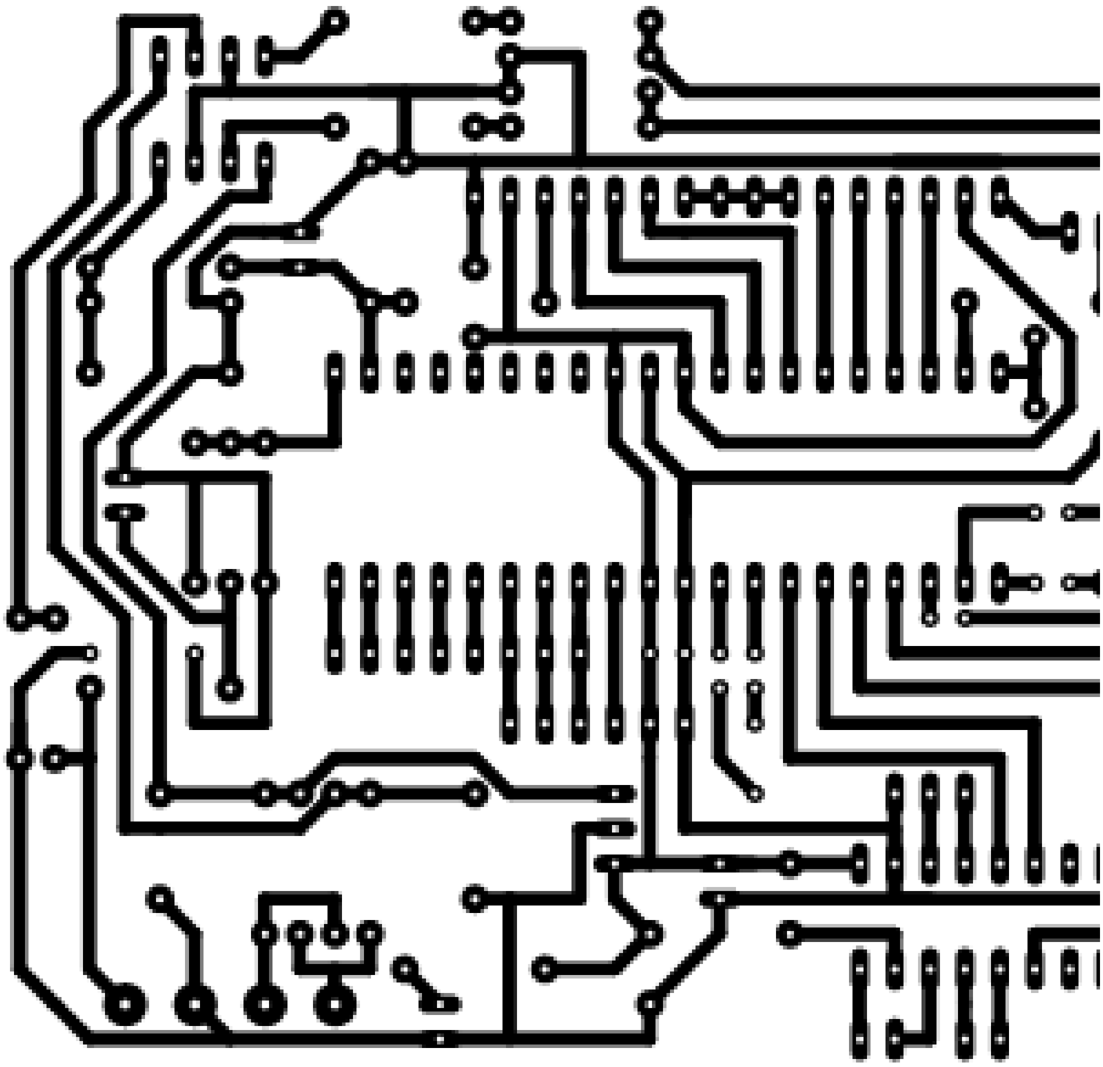
By integrating the entire design process, Circuit Wizard provides all the tools necessary to produce an electronics project from start to finish – even including on-screen testing of the PCB prior to construction.



The layout of the pcb was done using the same software.



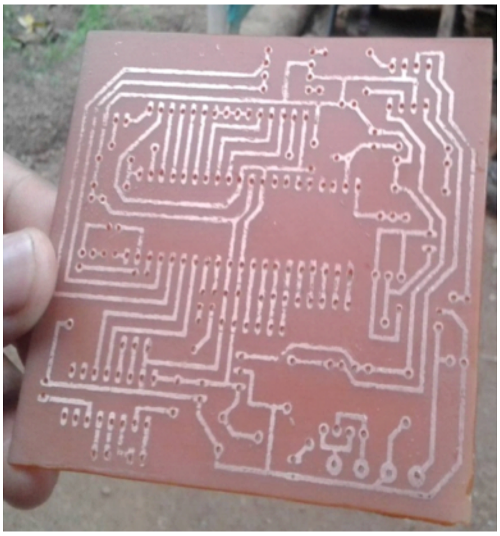


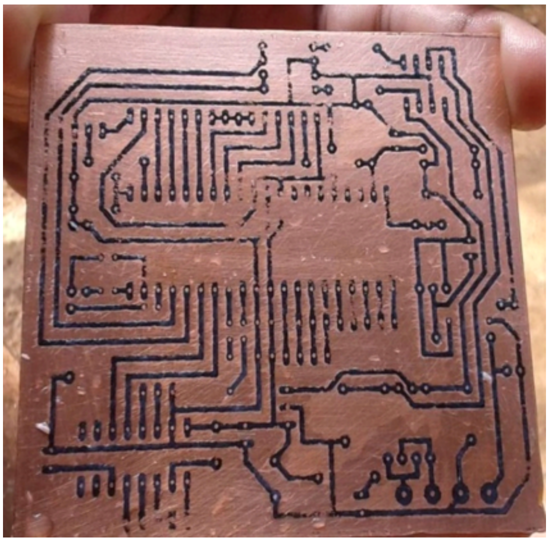


PCB etching, soldering and hardware finalisation

Etching of the PCB was done using toner transfer method as shown below:

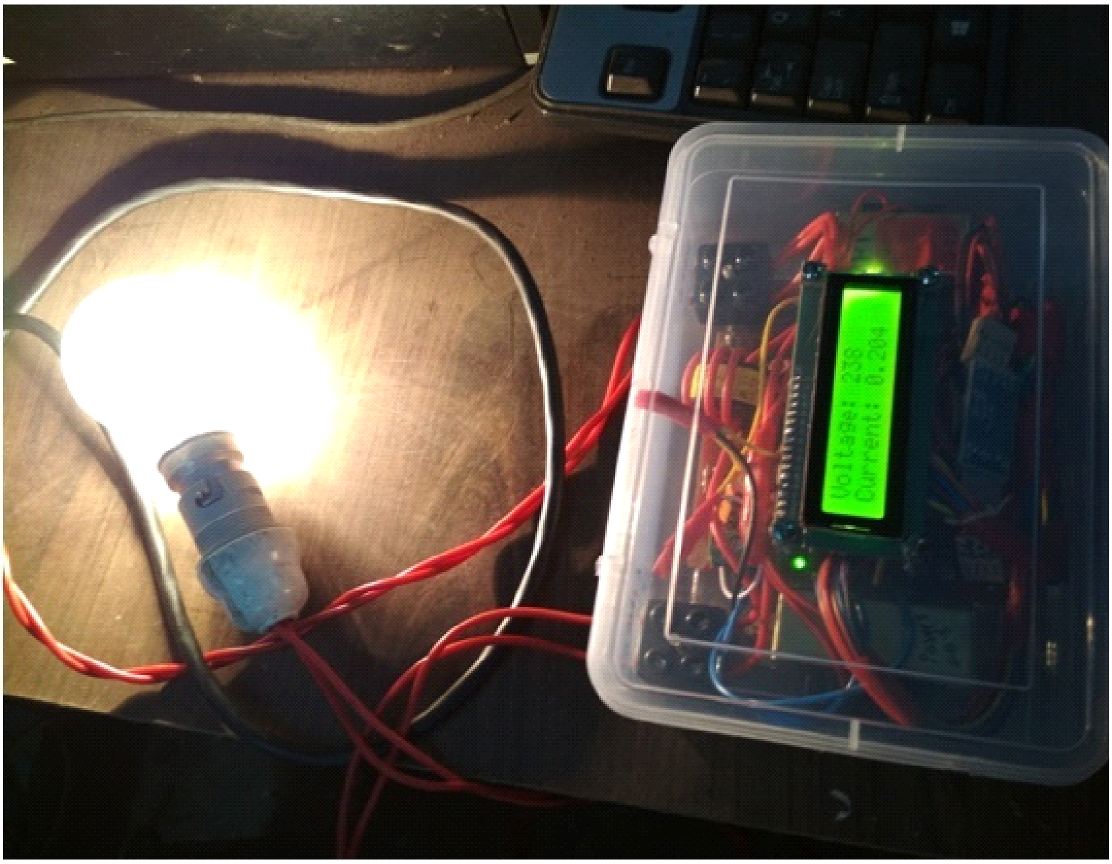
- Obtain a print of the PCB layout on a glossy paper using laser printer.
- Cut-out the required area of copper clad board, clean the surface using scrubber to remove oxide layer.
- Place the print on the copper clad board print side down and apply heat using a iron box for about 5 min.
- Remove the paper which has now stuck to the copper clad board by gently rubbing it using fingers only under running water.
- Now the toner of the print is transferred to the copper board.
- Place the board in ferric chloride solution (etchant) to remove copper from unwanted areas.
- After about 1 hr (depending on the size) the board will be ready. It is taken out and toner is removed by scrubbing gently



 Board after etching

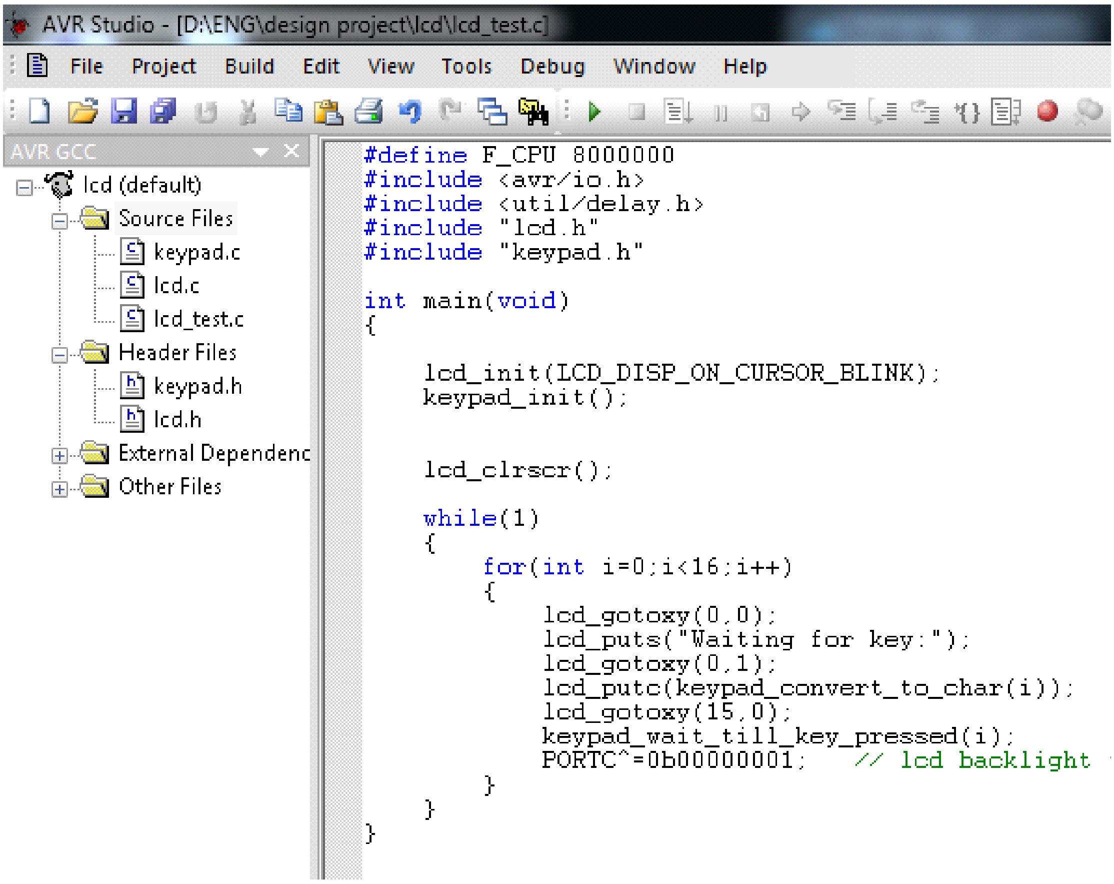
Board after toner transfer

The board thus obtained was drilled for the through-hole components. Components were placed and soldered. The only off board components were potential transformer and power supply transformer which were connected through terminal blocks connected on the PCB. All of these were placed in a suitable plastic container (cuboid box) and LINE OUT and LINE OUT terminals were provided. The entire hardware was complete with this step.



Software Design and testing

The software for the product was written using Atmel STUDIO 4.19. Atmel Studio 7 is the integrated development platform (IDP) for developing and debugging Atmel® SMART ARM® -based and Atmel AVR® microcontroller (MCU) applications. Studio 7 supports all AVR and Atmel SMART MCUs. The Atmel Studio 7 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code. It also connects seamlessly to Atmel debuggers and development kits.



The complier used was AVR-GCC-7.2.0-x86-mingw.

Various third open source party libraries were used these include:-

- Avrlibc
- Lcd library by Peter Fleury
- UART library by peter Fleury

The software implementation was done in stages rather than in a single step. Initially the LCD module was tested. In the next stage Voltage and frequency measurement part was added. Then current and PF measurement was added. Then energy calculations were implemented as the next stage. After this the meter readings were calibrated and only after calibration was the UART communication added. Tariff based calculations and notifications were implemented as the final step.

TESTING

For testing and calibration various other devices were used. These include:-

- Digital multimeter
- Oscilloscope
- 200w, 100w, 60w UPF lamp load
- Induction motor starting capacitance of 2.5μF.

The energy meter was calibrated against a industrial standard multimeter and a standard 200 watts UPF lamb load.

The meter was tested at 60W, 100W, 200W at UPF and LPF loading conditions. A keypad library was used additionally for testing purpose and is not included in the final product.

SOURCE CODE

```
#define F_CPU 8000000

#define UART_BAUD_RATE      9600

#define PF_LIMIT  0.700

#define C_LIMIT   0.7

#define U_LIMIT   0.01

#define TARRIF    3.25


#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include <avr/pgmspace.h>

#include <math.h>

#include <avr/eeprom.h>
```

```
#include "lcd.h"

#include "keypad.h"

#include "uart.h"


int var=0;

int sec=0;

int min=0;

int pf_flag=1;

int ma_flag=1;

int ta_flag=1;


int voltage_adc=0;

int voltage=0;

int current_adc=0;

float current=0;

int vflag=0;

float power=0;

float unit=0;

float bill=0;

int v_timer=0;

float v_frequency;

int i_timer;

float pf=0;

int ZCS_flag=0;


ISR(INT0_vect)

{

        if(ZCS_flag==0)i_timer=0;

        ZCS_flag=0;

        v_timer=TCNT1;

        TCNT1=0x0000; //reset timer
```

```
}


ISR(INT1_vect)
{
	ZCS_flag=1;
	i_timer=TCNT1;
}

ISR(ADC_vect)
{
	if(vflag)
  {
		voltage_adc=ADC;
		ADMUX |=0b00000001;//set ADC 1 for current
		vflag=0;		//clear voltage adc flag
		ADCSRA|=0b01000000;//start adc current conversion
	}
	else
	{
		current_adc=ADC;
		ADMUX&=0b11100000; //ADC 0 for voltage
	}

}



ISR(TIMER0_OVF_vect,ISR_NOBLOCK)
{
	var++;
```

```
if(var>30)
{
        var=0;
        sec++;
        if(sec>59)
        {
                sec=0;
                min++;
        }

        vflag=1;                        //set voltage adc  flag
        ADCSRA|=0b01000000; //start adc voltage conversion

        v_frequency=(float)(1000000.00/v_timer);
        voltage=(int)((float)voltage_adc*0.281);
        current=current_adc*0.00243;

        pf=cos( (float)6.28315 * (  (float)i_timer/(float)v_timer  ) );

        power=voltage*current*pf;
        unit+=(power/3600000.00);
        bill=unit*TARRIF;

        switch(sec%12)
        {
        case 0:
        case 1:
        case 2:{
                        lcd_clrscr();
                        lcd_puts("Voltage: ");
                        lcd_putc(keypad_convert_to_char(digit(voltage,3)));
                        lcd_putc(keypad_convert_to_char(digit(voltage,2)));
```

```
                lcd_putc(keypad_convert_to_char(digit(voltage,1)));
                lcd_gotoxy(0,1);
                lcd_puts("Current: ");
                lcd_putc(keypad_convert_to_char(digit(current,1)));
                lcd_putc('.');
                lcd_putc(keypad_convert_to_char(digit(current,-1)));
                lcd_putc(keypad_convert_to_char(digit(current,-2)));
                lcd_putc(keypad_convert_to_char(digit(current,-3)));
                break;
        }
case 3:
case 4:
case 5:{
                lcd_clrscr();
                lcd_puts("PF: ");
                lcd_putc(keypad_convert_to_char(digit(pf,1)));
                lcd_putc('.');
                lcd_putc(keypad_convert_to_char(digit(pf,-1)));
                lcd_putc(keypad_convert_to_char(digit(pf,-2)));
                lcd_putc(keypad_convert_to_char(digit(pf,-3)));
                lcd_gotoxy(0,1);
                lcd_puts("FREQ: ");
                lcd_putc(keypad_convert_to_char(digit(v_frequency,2)));
                lcd_putc(keypad_convert_to_char(digit(v_frequency,1)));
                lcd_putc('.');
                lcd_putc(keypad_convert_to_char(digit(v_frequency,-1)));
                lcd_putc(keypad_convert_to_char(digit(v_frequency,-2)));
                lcd_putc(keypad_convert_to_char(digit(v_frequency,-3)));
                break;
        }
case 6:
case 7:
```

```
case 8:{
        lcd_clrscr();
        lcd_puts("POWER: ");
        lcd_putc(keypad_convert_to_char(digit(power,3)));
        lcd_putc(keypad_convert_to_char(digit(power,2)));
        lcd_putc(keypad_convert_to_char(digit(power,1)));
        lcd_putc('.');
        lcd_putc(keypad_convert_to_char(digit(power,-1)));
        lcd_putc(keypad_convert_to_char(digit(power,-2)));
        lcd_putc(keypad_convert_to_char(digit(power,-3)));
        lcd_gotoxy(0,1);
        lcd_puts("Units: ");
        lcd_putc(keypad_convert_to_char(digit(unit,3)));
        lcd_putc(keypad_convert_to_char(digit(unit,2)));
        lcd_putc(keypad_convert_to_char(digit(unit,1)));
        lcd_putc('.');
        lcd_putc(keypad_convert_to_char(digit(unit,-1)));
        lcd_putc(keypad_convert_to_char(digit(unit,-2)));
        lcd_putc(keypad_convert_to_char(digit(unit,-3)));
        break;
    }
case  9:
case 10:
case 11:
        {
        lcd_clrscr();
        lcd_puts("BILL : ");
        lcd_putc(keypad_convert_to_char(digit(bill,3)));
        lcd_putc(keypad_convert_to_char(digit(bill,2)));
        lcd_putc(keypad_convert_to_char(digit(bill,1)));
        lcd_putc('.');
        lcd_putc(keypad_convert_to_char(digit(bill,-1)));
```

```
                                        lcd_putc(keypad_convert_to_char(digit(bill,-2)));
                                        lcd_putc(keypad_convert_to_char(digit(bill,-3)));
                                        lcd_gotoxy(0,1);
                                        lcd_puts("Tarrif:");
                                        lcd_putc(keypad_convert_to_char(digit(TARRIF,1)));
                                        lcd_putc('.');
                                        lcd_putc(keypad_convert_to_char(digit(TARRIF,-1)));
                                        lcd_putc(keypad_convert_to_char(digit(TARRIF,-2)));
                                        lcd_puts("/Kwh");
                                        break;
                                        }
                        }

                        if(fabs(pf)<PF_LIMIT&&pf_flag){         uart_puts("\n\rWARNING: Low power
factor!\n\r");                                  pf_flag=0;              }
                        else if     (fabs(pf)>PF_LIMIT)
                                    pf_flag=1;

                        if(current>C_LIMIT&&ma_flag)  {         uart_puts("\n\rWARNING: Max Connected
load exceeded!\n\r");                   ma_flag=0;              }
                        else if     (current<C_LIMIT)
                        ma_flag=1;

                        if(unit>U_LIMIT&&ta_flag)     {         uart_puts("\n\rWARNING: Subsidy usage
limit exceeded!\n\r");          ta_flag=0;              }
                        else if     (unit<U_LIMIT)
                                    ta_flag=1;

            }
}
```

```
int main(void)
{
		uart_init( UART_BAUD_SELECT(UART_BAUD_RATE,F_CPU) );
		sei();
		lcd_init(LCD_DISP_ON);
		keypad_init();
	PORTC|=0b00000001;
	lcd_clrscr();
		lcd_puts("LOADING....\nv1.8");
		_delay_ms(1000);
		uart_puts("\n\rDevice Online\n\r");

		DDRD &= 0b01111111;
		PORTD|= 0b10000000;

		TCCR1A=0b00000000;
		TCCR1B=0b00000010; //8 prescaler timer 1
		TCCR0 =0b00000101; //1024 prescaler timer 0
		TIMSK|=0b00000001; //interuppt on overflow Timer 0

		ADMUX =0b01000000; //ADC 0 for voltage
		ADCSRA=0b10001111; //INTERUPPT ENABLE-128_prescaler

		MCUCR|=0b00001111; //INT0,INT1 on rising edge
		GICR |=0b11000000; //INT0,INT1 enable

		char ch;
		unsigned int c;
		while(1)
		{
```

```
                    c = uart_getc();
        if ( c & UART_NO_DATA )
        {


        }
        else
        {
          if ( c & UART_FRAME_ERROR )
          uart_puts_P("UART Frame Error:\n\r ");
          if ( c & UART_OVERRUN_ERROR )
          uart_puts_P("UART Overrun Error:\n\r ");
          if ( c & UART_BUFFER_OVERFLOW )
          uart_puts_P("Buffer overflow error:\n\r ");


                            ch= ( 0x00FF&c);


                            if(ch=='f'||ch=='F') {          unit=0;    uart_puts("\n\rMemory flushed\n
\r");       }
                            if(ch=='c'||ch=='C')            uart_puts("\n\rDevice Online\n\r");
                            if(ch=='r'||ch=='R')
                            {
                                    uart_puts_P("\n\rPOWER: ");
                                    uart_putc(keypad_convert_to_char(digit(power,3)));
                                    uart_putc(keypad_convert_to_char(digit(power,2)));
                                    uart_putc(keypad_convert_to_char(digit(power,1)));
                                    uart_putc('.');
                                    uart_putc(keypad_convert_to_char(digit(power,-1)));
                                    uart_putc(keypad_convert_to_char(digit(power,-2)));
                                    uart_putc(keypad_convert_to_char(digit(power,-3)));


                                    uart_puts_P("\n\rUNITS: ");
```

```
uart_putc(keypad_convert_to_char(digit(unit,3)));
uart_putc(keypad_convert_to_char(digit(unit,2)));
uart_putc(keypad_convert_to_char(digit(unit,1)));
uart_putc('.');
uart_putc(keypad_convert_to_char(digit(unit,-1)));
uart_putc(keypad_convert_to_char(digit(unit,-2)));
uart_putc(keypad_convert_to_char(digit(unit,-3)));

uart_puts_P("\n\rVOLTAGE: ");
uart_putc(keypad_convert_to_char(digit(voltage,3)));
uart_putc(keypad_convert_to_char(digit(voltage,2)));
uart_putc(keypad_convert_to_char(digit(voltage,1)));

uart_puts_P("\n\rCURRENT: ");
uart_putc(keypad_convert_to_char(digit(current,1)));
uart_putc('.');
uart_putc(keypad_convert_to_char(digit(current,-1)));
uart_putc(keypad_convert_to_char(digit(current,-2)));
uart_putc(keypad_convert_to_char(digit(current,-3)));

uart_puts_P("\n\rFREQUENCY: ");
uart_putc(keypad_convert_to_char(digit(v_frequency,2)));
uart_putc(keypad_convert_to_char(digit(v_frequency,1)));

uart_putc('.');
uart_putc(keypad_convert_to_char(digit(v_frequency,-1)));
uart_putc(keypad_convert_to_char(digit(v_frequency,-2)));
uart_putc(keypad_convert_to_char(digit(v_frequency,-3)));

uart_puts_P("\n\rPOWER FACTOR: ");
uart_putc(keypad_convert_to_char(digit(pf,1)));
uart_putc('.');
```

```
                                        uart_putc(keypad_convert_to_char(digit(pf,-1)));
                                        uart_putc(keypad_convert_to_char(digit(pf,-2)));
                                        uart_putc(keypad_convert_to_char(digit(pf,-3)));


                                        uart_puts_P("\n\rBILL AMOUNT: ");
                                        uart_putc(keypad_convert_to_char(digit(bill,3)));
                                        uart_putc(keypad_convert_to_char(digit(bill,2)));
                                        uart_putc(keypad_convert_to_char(digit(bill,1)));
                                        uart_putc('.');
                                        uart_putc(keypad_convert_to_char(digit(bill,-1)));
                                        uart_putc(keypad_convert_to_char(digit(bill,-2)));
                                        uart_putc(keypad_convert_to_char(digit(bill,-3)));
                                        uart_puts_P("\n\rTarrif: ");
                                        uart_putc(keypad_convert_to_char(digit(TARRIF,1)));
                                        uart_putc('.');
                                        uart_putc(keypad_convert_to_char(digit(TARRIF,-1)));
                                        uart_putc(keypad_convert_to_char(digit(TARRIF,-2)));
                                        uart_putc(keypad_convert_to_char(digit(TARRIF,-3)));
                                        uart_puts_P("/UNIT");
                }
                                ch=0;
                        }
                }
        }
```

RESULT

The GSM Enabled smart digital energy meter was implemented. All desired output was obtained and verified against the expectations. Slight modifications was done on the objective and additional functionalities added as recommended by experts. Worst case scenarios were tested and meter working range, temperature, and other external factors were obtained and specified. Meter was designed for a range of 0-400 Volts and 0- 15 A. It was observed that the meter gave best results at range of 0-330 volts and 0 - 7 Amps. The temperature range was found by taking each component range into consideration using respective datasheets. The final working conditions after considering the factor of safety is as follows:

Voltage range: 0-300volts (RMS)

Current: 0-5Amps (RMS)

Frequency: 50Hz , 10% toleration

Temperature: 273K- 233K

Conclusion

The project was undertaken as part of the EEE 5 th semester course on Design project under the guidance of Prof. Jose Sebastian. Help of various sorts, both academic and intellectual was received from colleagues and faculty. The product was finally submitted for evaluation on 22.11.17

and was approved as successful.

References

- ATMEGA32 DATASHEET-Atmel group of microcontrollers.AVR
- LM358 datasheet*(National Semiconductor)" (PDF). Texas Instruments. 2000. Retrieved 4 Nov 2013.*
- MAX3232E (MAX3222E to MAX3246E) product webpage; Maxim.
- *AVR Microcontroller and Embedded Systems: Using Assembly and C*; Muhammad Ali Mazidi, Sarmad Naimi, Sepehr Naimi; 792 pages; 2010; ISBN 978-0138003319
- *"Measurement Canada Standard Dwg. No.3400 D3 Delta Connected CTs" (PDF). MEASUREMENT CANADA. Retrieved 12 December 2012.*

- *"Limits of the 0.3 and 0.6 accuracy classes for measuring current transformers" . Measurement Canada. Retrieved 18 April 2013.*
- *"PS-E-15 — Provisional Specifications for Approval of Electronic Voltage Transformers". Measurement Canada. Retrieved 18 April 2013.*

- *"PS-E-15 — Provisional Specifications for Approval of Electronic Voltage Transformers". Measurement Canada. Retrieved 18 April 2013.*