```
// SOLAR CHARGE CONTROLLER
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define SOL_ADC A0     // Solar panel side voltage divider is connected to pin A0
#define BAT_ADC A1    // Battery side voltage divider is connected to pin A1
#define CURRENT_ADC A2  // ACS 712 current sensor is connected to pin A2
#define TEMP_ADC A3   // LM 35 Temperature is connected to pin A3
#define AVG_NUM 10    // number of iterations of the adc routine to average the adc readings
#define BAT_MIN 10.5  // minimum battery voltage for 12V system
#define BAT_MAX 15.0  // maximum battery voltage for 12V system
#define BULK_CH_SP 14.4 // bulk charge set point for sealed lead acid battery // flooded type set it to
14.6V
#define FLOAT_CH_SP 13.6  //float charge set point for lead acid battery
#define LVD 11.5          //Low voltage disconnect setting for a 12V system
#define PWM_PIN 3         // pin-3 is used to control the charging MOSFET //the default frequency is
490.20Hz
#define LOAD_PIN 2       // pin-2 is used to control the load
#define BAT_RED_LED 5
#define BAT_GREEN_LED 6
#define BAT_BLUE_LED 7
#define LOAD_RED_LED 8
#define LOAD_GREEN_LED 9
//------------------------------------------------------------------------------------------------------------------------------
//////////////////DECLARATION OF ALL BIT MAP ARRAY FOR FONTS////////////////////////////////////////
//------------------------------------------------------------------------------------------------------------------------------

byte solar[8] = //icon for solar panel
{
  0b11111,0b10101,0b11111,0b10101,0b11111,0b10101,0b11111,0b00000
};
byte battery[8] =  //icon for battery
{
  0b01110,0b11011,0b10001,0b10001,0b10001,0b10001,0b10001,0b11111
};

byte energy[8] =  // icon for power
{
  0b00010,0b00100,0b01000,0b11111,0b00010,0b00100,0b01000,0b00000
};
/*byte alarm[8] =  // icon for alarm
{
 0b00000,0b00100,0b01110,0b01110,0b01110,0b11111,0b00000,0b00100
};*/
byte temp[8] = //icon for termometer
{
 0b00100,0b01010,0b01010,0b01110,0b01110,0b11111,0b11111,0b01110
};

byte charge[8] = // icon for battery charge
{
  0b01010,0b11111,0b10001,0b10001,0b10001,0b01110,0b00100,0b00100,
};
byte not_charge[8]=
{
  0b00000,0b10001,0b01010,0b00100,0b01010,0b10001,0b00000,0b00000,
};
```

```
//------------------------------------------------------------------------------------------------------------------------
///////////////////////DECLARATION OF ALL GLOBAL VARIABLES//////////////////////////////////////////////
//------------------------------------------------------------------------------------------------------------------------

float solar_volt=0;

float bat_volt=0;

float load_current=0;

int temperature=0;

int temp_change=0;

float system_volt=0;

float bulk_charge_sp=0;

float float_charge_sp=0;

float charge_status=0;

float load_status=0;

float error=0;

float Ep=0;

int duty =0;

float lvd;

float msec=0;

float last_msec=0;

float elasped_msec=0;

float elasped_time=0;

float ampSecs = 0;

float ampHours=0;

float watts=0;

float wattSecs = 0;

float wattHours=0;


// Set the pins on the I2C chip used for LCD connections:

//                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
```

```
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // Set the LCD I2C address // In my case 0x27

//**************************************** MAIN PROGRAM START ***************************

void setup()

{

Serial.begin(9600);

pinMode(BAT_RED_LED,OUTPUT);

pinMode(BAT_GREEN_LED,OUTPUT);

pinMode(BAT_BLUE_LED,OUTPUT);

pinMode(LOAD_RED_LED ,OUTPUT);

pinMode(LOAD_GREEN_LED,OUTPUT);

pinMode(PWM_PIN,OUTPUT);

pinMode(LOAD_PIN,OUTPUT);

digitalWrite(PWM_PIN,LOW);  // default value of pwm duty cycle

digitalWrite(LOAD_PIN,LOW);  // default load state is OFF

lcd.begin(20,4);   // initialize the lcd for 16 chars 2 lines, turn on backlight

lcd.backlight(); // finish with backlight on

lcd.createChar(1,solar);

lcd.createChar(2, battery);

lcd.createChar(3, energy);

//lcd.createChar(4,alarm);

lcd.createChar(5,temp);

lcd.createChar(6,charge);

lcd.createChar(7,not_charge);

lcd.clear();

}


void loop()

{

 read_data();            // read different sensors data from analog pin of arduino
 system_voltage();       // detect the system voltage according to battery voltage
 setpoint();     // decide the charge set point according to system voltage
 charge_cycle();        // pwm charging of battery
 power();              // calculate the load power and energy
 load_control();        //control the load
 led_indication();      // led indica
 print_data();          // print in serial monitor
 lcd_display();          // lcd display
 }
//***************************************** PROGRAM END *************************


//--------------------------------------------------------------------------------------------------
///////// READS AND AVERAGES THE ANALOG INPUTS (SOLRAR VOLTAGE,BATTERY VOLTAGE)////// ///
//--------------------------------------------------------------------------------------------------
int read_adc(int adc_parameter)
{
   int sum = 0;
  int sample ;
  for (int i=0; i<AVG_NUM; i++)
  {                                  // loop through reading raw adc values AVG_NUM number of times
    sample = analogRead(adc_parameter);    // read the input pin
    sum += sample;                       // store sum for averaging
    delayMicroseconds(50);             // pauses for 50 microseconds
  }
  return(sum / AVG_NUM);              // divide sum by AVG_NUM to get average and return it
}
//---------------------------------------------------------------------------------------------------------
/////////////////////READ THE DATA///////////////////////////////////////////////////////////
//---------------------------------------------------------------------------------------------------------
```

```
 void read_data(void)

 {

   //5V = ADC value 1024 => 1 ADC value = (5/1024)Volt= 0.0048828Volt

   // Vout=Vin*R2/(R1+R2) => Vin = Vout*(R1+R2)/R2   R1=100 and R2=20

    solar_volt = read_adc(SOL_ADC)*0.00488*(120/20);

    bat_volt   = read_adc(BAT_ADC)*0.00488*(120/20);

    load_current = (read_adc(CURRENT_ADC)*.0488 -25);

    temperature = read_adc(TEMP_ADC)*0.00488*100;

      }

  //----------------------------------------------------------------------------------------------------------

/////////////////////////////////POWER AND ENERGY CALCULATION //////////////////////////////////

//----------------------------------------------------------------------------------------------------------

void power(void)

{

msec = millis();

elasped_msec = msec - last_msec; //Calculate how long has past since last call of this function

elasped_time = elasped_msec / 1000.0; // 1sec=1000 msec

watts = load_current * bat_volt; //Watts now

ampSecs = (load_current*elasped_time); //AmpSecs since last measurement

wattSecs = ampSecs * bat_volt; //WattSecs since last measurement

ampHours = ampHours + ampSecs/3600; // 1 hour=3600sec //Total ampHours since program started

wattHours = wattHours + wattSecs/3600; // 1 hour=3600sec //Total wattHours since program started

last_msec = msec; //Store 'now' for next time

}

//----------------------------------------------------------------------------------------------------------

////////////////////////////////////PRINT DATA IN SERIAL MONITOR//////////////////////////////

//----------------------------------------------------------------------------------------------------------

  void print_data(void)

  {
```

```
delay(100);

Serial.print("Solar Panel Voltage: ");

Serial.print(solar_volt);

Serial.println("V");

Serial.print("Battery Voltage: ");

Serial.print(bat_volt);

Serial.println("V");

Serial.print("Syestem Voltage: ");

Serial.print(system_volt);

Serial.println("V");

Serial.print("Charge Set Point:");

Serial.println(bulk_charge_sp);

Serial.print("Temperature:");

Serial.print(temperature);

Serial.println("C");

Serial.print("Load Current: ");

Serial.print(load_current);

Serial.println("A");

Serial.print("Power: ");

Serial.print(watts);

Serial.println("W");

Serial.print("Energy: ");

Serial.print(wattHours);

Serial.println("WH");

Serial.print("Duty Cycle :");

if (charge_status==1)

{

Serial.println("99%");

Serial.println("BULK CHARGING");
```

```
    }
    else if (charge_status==2)
    {
    Serial.print(Ep);
    Serial.println("%");
    Serial.println("FLOAT CHARGING");
    }
    else
    {
    Serial.println("0%");
    Serial.println("NOT CHARGING");
    }
    if(load_status==1)
    {
     Serial.println("LOAD IS CONNECTED");
    }
    else
    {
     Serial.println("LOAD IS DISCONNECTED");
    }

    Serial.println("***************************");
 }
//-------------------------------------------------------------------------------------------------------------------
////////////////////////SYSTEM VOLTAGE AUTO DETECT ///////////////////////////////////////////////
//-------------------------------------------------------------------------------------------------------------------
void system_voltage(void)
{
  if ((bat_volt >BAT_MIN) && (bat_volt < BAT_MAX))
  {
     system_volt = 12;
  }
  /*
  else if  ((bat_volt > BAT_MIN*2 ) && (bat_volt < BAT_MAX*2))
  {
    system_volt=24;
  }*/
  else if ((bat_volt > BAT_MIN/2 ) && (bat_volt < BAT_MAX/2))
  {
    system_volt=6;
  }

}
//------------------------------------------------------------------------------------------------------------------------
 /////////////////////////CHARGE SET POINT ///////////////////////////////////////////////////////
//------------------------------------------------------------------------------------------------------------------------

void setpoint(void)
{
  temp_change =temperature-25.0; // 25deg cel is taken as standard room temperature
 // temperature compensation = -5mv/degC/Cell
  // If temperature is above the room temp ;Charge set point should reduced
  // If temperature is bellow the room temp ;Charge set point should increased
  if(system_volt ==12)
  {
     bulk_charge_sp = BULK_CH_SP-(0.030*temp_change) ;
     float_charge_sp=FLOAT_CH_SP-(0.030*temp_change) ;
     lvd =LVD;
  }


  else if(system_volt ==6)

  {

     bulk_charge_sp = (BULK_CH_SP/2)-(0.015*temp_change) ;

     float_charge_sp= (FLOAT_CH_SP/2)-(0.015*temp_change) ;

     lvd=LVD/2;

  }

  /*

  else if (system_volt == 24)

  {

   bulk_charge_sp = (BULK_CH_SP*2)-(0.060*temp_change) ;

   float_charge_sp= (FLOAT_CH_SP*2)-(0.060*temp_change) ;

   lvd=LVD*2;

  }

  */


}

//-----------------------------------------------------------------------------------------------------------------------------

  ////////////////////////////////////PWM CHARGE CYCLE @500 HZ /////////////////////////////////////////

 //----------------------------------------------------------------------------------------------------------------------------

void charge_cycle(void)

{

  if (solar_volt > bat_volt && bat_volt <= bulk_charge_sp)

  {


   if (bat_volt <= float_charge_sp) // charging start

  {

     charge_status = 1; // indicate the charger is in BULK mode
     duty= 252.45;
     analogWrite(PWM_PIN,duty); // 99 % duty cycle // rapid charging


  }
  else if (bat_volt >float_charge_sp && bat_volt <= bulk_charge_sp)
  {
     charge_status = 2; // indicate the charger is in FLOAT mode
     error  = (bulk_charge_sp - bat_volt);      // duty cycle reduced when the battery voltage approaches the charge set point
     Ep= error *100 ; //Ep= error* Kp // Assume  Kp=100


     if(Ep < 0)
      {
       Ep=0;
       }
     else if(Ep>100)
       {
        Ep=100;
       }
     else if(Ep>0 && Ep <=100) // regulating
      {
        duty = (Ep*255)/100;
      }
      analogWrite(PWM_PIN,duty);
   }
}
   else
   {
   charge_status=0;  // indicate the charger is OFF
```

```
   duty=0;
   analogWrite(PWM_PIN,duty);
   }
}
//-----------------------------------------------------------------------------------------------------------------
///////////////////////////////////LOAD CONTROL/////////////////////////////////////////////////////
//-----------------------------------------------------------------------------------------------------------------

void load_control()
{
 if (solar_volt < 5  ) // load will on when night
{
  if(bat_volt >lvd)   // check if battery is healthy
  {
  load_status=1;
  digitalWrite(LOAD_PIN, HIGH); // load is ON
  }
  else if(bat_volt < lvd)
  {
    load_status=0;
   digitalWrite(LOAD_PIN, LOW); //load is OFF
  }
 }
 else // load will off during day
 {
   load_status=0;
   digitalWrite(LOAD_PIN, LOW);
 }
}
```

```
//-------------------------------------------------------------------------------------------------
//////////////////////////LED INDICATION////////////////////////////////////
//-------------------------------------------------------------------------------------------------
void led_indication(void)
{
  battery_led();          //Battery status led indication
  load_led();             //Load led indication
}

//-----------------------------------------------------------------------------------------------------------------------------
///////////////////////////////////////////////BATTERY LED INDICATION//////////////////////////////////////
//-----------------------------------------------------------------------------------------------------------------------------
void battery_led(void)
{

   if( (bat_volt > system_volt) && ( bat_volt <bulk_charge_sp))
  {
      leds_off_all();
      digitalWrite(BAT_GREEN_LED,LOW);  // battery voltage is healthy
  }
  else if(bat_volt >= bulk_charge_sp)
  {
      leds_off_all();
      digitalWrite(BAT_BLUE_LED,LOW);  //battery is fully charged
  }
   else if(bat_volt < system_volt)
  {
      leds_off_all();
      digitalWrite(BAT_RED_LED,LOW);  // battery voltage low
  }
}
//--------------------------------------------------------------------------------------------------------------------
/////////////////////////////////////LOAD LED INDICATION////////////////////////////////////////////////
//--------------------------------------------------------------------------------------------------------------------


  void load_led()
  {
    if(load_status==1)
    {
      digitalWrite(LOAD_GREEN_LED,HIGH);
    }
    else if(load_status==0)
    {
      digitalWrite(LOAD_RED_LED,HIGH);
    }
   }

//----------------------------------------------------------------------------------------------------
////////////////////////// TURN OFF ALL THE LED/////////////////////////////////////////////////////////////////
//----------------------------------------------------------------------------------------------------
void leds_off_all(void)
{
    digitalWrite(BAT_RED_LED,HIGH);
  digitalWrite(BAT_GREEN_LED,HIGH);
  digitalWrite(BAT_BLUE_LED,HIGH);
  digitalWrite(LOAD_RED_LED, LOW);
  digitalWrite(LOAD_GREEN_LED, LOW);
}
```

```
//--------------------------------------------------------------------------------------------------
//////////////////////// LCD DISPLAY///////////////////////////////////////////////////////////
//--------------------------------------------------------------------------------------------------
void lcd_display()
{
 lcd.setCursor(0, 0);
 lcd.write(1);
 lcd.setCursor(2, 0);
 lcd.print(solar_volt);
 lcd.print("V");
 lcd.setCursor(14, 0);
 lcd.write(5);
 lcd.setCursor(16, 0);
 lcd.print(temperature);
 lcd.write(0b11011111);
 lcd.print("C");
 lcd.setCursor(0,1);
 lcd.write(2);
 lcd.setCursor(2, 1);
 lcd.print(bat_volt);
 lcd.print("V");
 lcd.setCursor(14, 1);
 lcd.write(2);
 if((charge_status==1) | (charge_status== 2))
 {
 lcd.write(6);
 }
 else
 {
```

```
 lcd.write(7);
 }
 lcd.setCursor(0,2);
 lcd.write(3);
 lcd.setCursor(2,2);
 lcd.print(load_current);
 lcd.print("A");
 lcd.setCursor(13,2);
 lcd.print(watts);
 lcd.print("W");
 lcd.setCursor(0,3);
 lcd.print("Energy:");
 lcd.print(wattHours);
 lcd.print("WH");


}
```