# ATMEGA328 – Main_Code

```
//Only Change these Parameters if Required

String Meter_Number = "00001";
float Input_Voltage = 4.96;
int Page_Change_Delay = 4000;
int In_Page_Loop_Delay = 1000;
float Country_Frequency = 50.0;

////////Code Begins Here//////////////////

#include <LiquidCrystal_I2C.h>
#include "ADS1X15.h"
#include <Wire.h>
#include <SoftwareSerial.h>
#include <ZMPT101B.h>
#include "RTClib.h"
#include <EEPROM.h>

#define SENSITIVITY 500.0f

LiquidCrystal_I2C lcd(0X27, 20, 4); //(address, Length, Width)

SoftwareSerial s(4, 2);  //(RX,TX)

ADS1115 ADS(0x48);

//Voltage ZMPT101B
ZMPT101B VoltSense(A2, Country_Frequency);
float V = 0;

//Current ACS758
float I = 0;
uint8_t i = 0;
uint32_t period = 1000000 / Country_Frequency;

//Power
double realPower;
double Pfactor = 0.0;
float P = 0;

//Units & Price
float units = 0;
float Unit_Day, Unit_Eve, Unit_Night, Today_Unit;
```

```
int This_Day;

//Data Send 'n' Recieve
String PriceDay, PriceNight, PriceEve;
float fPriceDay = 0.0;
float fPriceNight = 0.0;
float fPriceEve = 0.0;
float Pri_Day = 0.0;
float Pri_Eve = 0.0;
float Pri_Night = 0.0;
float Total_Rate = 0.0;

char c;
String dataIn;
int8_t indexOfA, indexOfB, indexOfC;

//Initialize other Variables
long Screenxt = 0;
unsigned long lasttime = 0;
long ScreenSelect = 0;

//Time RTC
RTC_DS3231 rtc;

byte RupeeSymbol[] = { 0x1F, 0x02, 0x1F, 0x04, 0x08, 0x04, 0x02, 0x01};

void setup() {
  Serial.begin(115200);
  s.begin(9600);
  VoltSense.setSensitivity(SENSITIVITY);
  Wire.begin();
  rtc.begin();
  lcd.init();
  lcd.backlight();
  lcd.setCursor(5, 0);
  lcd.print("POWERTRACK");
  lcd.setCursor(2, 2);
  lcd.print("Meter No : ");
  lcd.print(Meter_Number);
  delay(3000);
  lcd.clear();

  lcd.createChar(1, RupeeSymbol);
```

```
  if (rtc.lostPower())  //sets the RTC to the date & time this sketch was
compiled
  {
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
  ADS.begin();

  //EEPROM Usage
  int k;
  EEPROM.get(0, k);

  if (k != 12) {
    EEPROM.put(0, 12);
    Unit_Day = 0.00f;
    Unit_Eve = 0.00f;
    Unit_Night = 0.00f;
    Today_Unit =0.00f;
    This_Day = 0;

  } else if (k = 12) {
    EEPROM.get(40, Today_Unit);
    EEPROM.get(50, Unit_Day);
    EEPROM.get(60, Unit_Eve);
    EEPROM.get(70, Unit_Night);
    EEPROM.get(90, This_Day);

  }

  delay(100);
}

void loop() {

  if (millis() - Screenxt > Page_Change_Delay) {
    ScreenSelect += 1;
    Screenxt = millis();
  }

  if (millis() - lasttime > 500) {
    if ((ScreenSelect % 3) == 1) {
      lcd.clear();
      displayPage_1();
    }

    else if ((ScreenSelect % 3) == 2) {
```

```
      lcd.clear();
      displayPage_2();
    }

    else if ((ScreenSelect % 3) == 0) {
      lcd.clear();
      displayPage_3();
    }
  }
}

void displayPage_1() {
  for (int j = 0; j < 5; j++) {

    lcd.setCursor(5, 0);
    lcd.print("POWERTRACK");

    time();

    Get_Voltage();

    getRmsCurrent();

    getPower();

    getEnergy();

    Price_Calculation();

    SendReceiveData();

    //Voltage
    if (V < 40) {
      lcd.setCursor(0, 2);
      lcd.print("V:000.00V ");  //Voltage

    } else {
      lcd.setCursor(0, 2);
      lcd.print("V:" + String(V, 2) + "V ");
    }

    //Current
    if (I < 0.031) {
      lcd.setCursor(0, 3);
      lcd.print("I:00.00A  ");  //Current
```

```
    } else if (I <= 9) {
      lcd.setCursor(0, 3);
      lcd.print("I:0" + String(I, 2) + "A");
    } else {
      lcd.setCursor(0, 3);
      lcd.print("I:" + String(I, 2) + "A");
    }

    //Power
    if (P >= 1000) {
      lcd.setCursor(10, 2);
      lcd.print("P:" + String(P / 1000, 3) + "kW");
    } else {
      lcd.setCursor(10, 2);
      lcd.print("P:" + String(P, 1) + "W");
    }

    //Power Factor
    if (Today_Unit >= 1000) {
      lcd.setCursor(10, 3);
      lcd.print("TU:" + String(Today_Unit/1000, 2) + "kWh");
    } else {
      lcd.setCursor(10, 3);
      lcd.print("TU:" + String(Today_Unit, 1) + "Wh");
    }

    delay(In_Page_Loop_Delay);
  }
  lasttime = millis();
}

void displayPage_2() {

  for (int j = 0; j < 5; j++) {

    lcd.setCursor(2, 0);
    lcd.print("UNITS wrt TIME");

    time();

    Get_Voltage();

    getRmsCurrent();

    getPower();
```

```
    getEnergy();

    Price_Calculation();

    SendReceiveData();

    lcd.setCursor(0, 2);
    lcd.print("D:" + String(Unit_Day, 2) + "kWh");

    lcd.setCursor(0, 3);
    lcd.print("E:" + String(Unit_Eve, 2) + "kWh");

    lcd.setCursor(10, 2);
    lcd.print("N:" + String(Unit_Night, 2) + "kWh");

    if (units >= 1) {
      lcd.setCursor(10, 3);
      lcd.print("U:" + String(units, 3) + "kWh");
    } else {
      lcd.setCursor(10, 3);
      lcd.print("U:" + String(units * 1000, 1) + "Wh");
    }

    delay(In_Page_Loop_Delay);
  }
  lasttime = millis();
}

void displayPage_3() {

  for (int j = 0; j < 5; j++) {

    lcd.setCursor(2, 0);
    lcd.print("ELECTRICITY BILL");

    time();

    Get_Voltage();

    getRmsCurrent();

    getPower();

    getEnergy();
```

```
    Price_Calculation();

    SendReceiveData();

    lcd.setCursor(0, 2);
    lcd.print("D:");
    lcd.write(1);
    lcd.print(Pri_Day);

    lcd.setCursor(0, 3);
    lcd.print("E:");
    lcd.write(1);
    lcd.print(Pri_Eve);

    lcd.setCursor(10, 2);
    lcd.print("N:");
    lcd.write(1);
    lcd.print(Pri_Night);

    lcd.setCursor(10, 3);
    lcd.print("T:");
    lcd.write(1);
    lcd.print(Total_Rate);

    delay(In_Page_Loop_Delay);
  }
  lasttime = millis();
}

void Parse_the_Data() {
  indexOfA = dataIn.indexOf("A");
  indexOfB = dataIn.indexOf("B");
  indexOfC = dataIn.indexOf("C");

  PriceDay = dataIn.substring(0, indexOfA);
  PriceEve = dataIn.substring(indexOfA + 1, indexOfB);
  PriceNight = dataIn.substring(indexOfB + 1, indexOfC);
}

void time() {
  DateTime now = rtc.now();
  lcd.setCursor(0, 1);
  lcd.print("Time:");
  lcd.setCursor(6, 1);
```

```
  if (now.hour() <= 9) {
    lcd.print("0");
    lcd.setCursor(7, 1);
    lcd.print(now.hour(), DEC);
  } else {
    lcd.print(now.hour(), DEC);
  }
  lcd.setCursor(8, 1);
  lcd.print(":");
  lcd.setCursor(9, 1);
  if (now.minute() <= 9) {
    lcd.print("0");
    lcd.setCursor(10, 1);
    lcd.print(now.minute(), DEC);
  } else {
    lcd.print(now.minute(), DEC);
  }
  lcd.setCursor(11, 1);
  lcd.print(":");
  lcd.setCursor(12, 1);

  if (now.second() <= 9) {
    lcd.print("0");
    lcd.setCursor(13, 1);
    lcd.print(now.second(), DEC);
  } else {
    lcd.print(now.second(), DEC);
  }
}

int getZeroPoint() {
  uint32_t Isum = 0;
  uint32_t measurements_count = 0;
  uint32_t t_start = micros();

  while (micros() - t_start < period) {
    Isum += ADS.readADC(0);
    measurements_count++;
  }

  return Isum / measurements_count;
}

void Get_Voltage() {
  V = VoltSense.getRmsVoltage();
```

```
  if (V < 40) {
    V = 0;
  }
}

void getRmsCurrent() {
  double readingCurrent = 0.0f;

  for (uint8_t i = 0; i < 2; i++) {
    int zeroPoint = getZeroPoint();

    int32_t Inow = 0;
    uint32_t Isum = 0;
    uint32_t measurements_count = 0;
    uint32_t t_start = micros();

    while (micros() - t_start < period) {
      Inow = ADS.readADC(0) - zeroPoint;
      Isum += (Inow * Inow);
      measurements_count++;
    }

    readingCurrent+=sqrt(Isum / measurements_count)/26667 * Input_Voltage /0.040;
  }

  I = readingCurrent / 2;

  if (I < 0.031) {
    I = 0;
  }
}

void getPower() {
  double Prms = 0.0;
  double realPowerT = 0;
  double _Prm = 0.0;

  for (uint8_t i = 0; i < 2; i++) {
    double Pnow = 0;
    double Psum = 0;
    double Ponlysum = 0;

    uint32_t measurements_count = 0;
    uint32_t t_start = micros();
    realPower = 0;
```

```
    while (micros() - t_start < period) {
      Get_Voltage();
      getRmsCurrent();

      Pnow = V * I;
      Ponlysum += Pnow;
      Psum += (Pnow * Pnow);
      measurements_count++;
    }

    realPowerT = (Ponlysum / measurements_count);

    Prms += sqrt(Psum / measurements_count);
  }

  P = Prms / 2;

  if (P == 0) {
    Pfactor = 0;
  } else {
    if ((realPowerT / P) >= 0.95) {
      Pfactor = 0.95;
    } else {
      Pfactor = realPowerT / P;
    }
  }
}

void getEnergy() {
  float elapsedTimeHours = 1.03 / 3600.0;

  DateTime now1 = rtc.now();

  if (now1.hour() >= 6 && now1.hour() < 18) {
    Unit_Day = Unit_Day + ((P / 1000) * elapsedTimeHours);
  } else if (now1.hour() >= 18 && now1.hour() < 22) {
    Unit_Eve = Unit_Eve + ((P / 1000) * elapsedTimeHours);
  } else {
    Unit_Night = Unit_Night + ((P / 1000) * elapsedTimeHours);
  }

  if(now1.day() == This_Day){
   Today_Unit = Today_Unit + (P * elapsedTimeHours);
  }
```

```
  else if(now1.day() != This_Day){
    Today_Unit = 0;
    This_Day = now1.day();
    EEPROM.put(90, This_Day);
  }

  EEPROM.put(40, Today_Unit);
  EEPROM.put(50, Unit_Day);
  EEPROM.put(60, Unit_Eve);
  EEPROM.put(70, Unit_Night);

  units = (Unit_Day + Unit_Eve + Unit_Night);
}

void Price_Calculation() {
  Pri_Day = fPriceDay * Unit_Day;

  Pri_Eve = fPriceEve * Unit_Eve;

  Pri_Night = fPriceNight * Unit_Night;

  Total_Rate = Pri_Day + Pri_Eve + Pri_Night;
}

void SendReceiveData() {
  while (s.available() > 0) {
    c = s.read();

    if (c == '\n') {
      break;
    } else {
      dataIn += c;
    }
  }

  if (c == '\n') {
    Parse_the_Data();

    fPriceDay = PriceDay.toFloat();
    fPriceEve = PriceEve.toFloat();
    fPriceNight = PriceNight.toFloat();

    c = 0;
    dataIn = "";
  }
```

```
  s.print(V, 2); s.print("A"); s.print(I, 2); s.print("B"); s.print(P / 1000, 3);
  s.print("C"); s.print(units, 3); s.print("D"); s.print(Total_Rate, 2);
  s.print("E"); s.print(Today_Unit,3); s.print("F"); s.print("\n");
  s.flush();
}
```