#define BLYNK\_PRINT Serial

#include "EmonLib.h"   //https://github.com/openenergymonitor/EmonLib

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>

EnergyMonitor emon;

#define vCalibration 106.8

#define currCalibration 0.52

BlynkTimer timer;

char auth[] = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

char ssid[] = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

char pass[] = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*3";

float kWh = 0;

unsigned long lastmillis = millis();

void myTimerEvent() {

    emon.calcVI(20, 2000);

    Serial.print("Vrms: ");

    Serial.print(emon.Vrms, 2);

    Serial.print("V");

    Blynk.virtualWrite(V0, emon.Vrms);

    Serial.print("\tIrms: ");

    Serial.print(emon.Irms, 4);

    Serial.print("A");

    Blynk.virtualWrite(V1, emon.Irms);

    Serial.print("\tPower: ");

    Serial.print(emon.apparentPower, 4);

    Serial.print("W");

    Blynk.virtualWrite(V2, emon.apparentPower);

    Serial.print("\tkWh: ");

    kWh = kWh + emon.apparentPower\*(millis()-lastmillis)/3600000000.0;

    Serial.print(kWh, 4);

    Serial.println("kWh");

    lastmillis = millis();

    Blynk.virtualWrite(V3, kWh);

}

void setup() {

  Serial.begin(9600);

  emon.voltage(35, vCalibration, 1.7); // Voltage: input pin, calibration, phase\_shift

  emon.current(34, currCalibration); // Current: input pin, calibration.

  Blynk.begin(auth, ssid, pass);

  timer.setInterval(5000L, myTimerEvent);

}

void loop() {

  Blynk.run();

  timer.run();

}

Source code:

#include <LiquidCrystal.h>

LiquidCrystal lcd(13, 12, 14, 27, 26, 25);

#define BLYNK\_PRINT Serial

#include "EmonLib.h"

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>

EnergyMonitor emon;

#define vCalibration 83.3

#define currCalibration 0.50

BlynkTimer timer;

char auth[] = "hsYG\_5da4gdP9jZkL18O5RNcJSrBT-Ou";

char ssid[] = "Alexahome";

char pass[] = "loranthus";

float kWh = 0;

unsigned long lastmillis = millis();

void myTimerEvent()

{

  emon.calcVI(20, 2000);

  kWh = kWh + emon.apparentPower \* (millis() - lastmillis) / 3600000000.0;

  yield();

  Serial.print("Vrms: ");

  Serial.print(emon.Vrms, 2);

  Serial.print("V");

  Serial.print("\tIrms: ");

  Serial.print(emon.Irms, 4);

  Serial.print("A");

  Serial.print("\tPower: ");

  Serial.print(emon.apparentPower, 4);

  Serial.print("W");

  Serial.print("\tkWh: ");

  Serial.print(kWh, 5);

  Serial.println("kWh");

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Vrms:");

  lcd.print(emon.Vrms, 2);

  lcd.print("V");

  lcd.setCursor(0, 1);

  lcd.print("Irms:");

  lcd.print(emon.Irms, 4);

  lcd.print("A");

  delay(2500);

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Power:");

  lcd.print(emon.apparentPower, 4);

  lcd.print("W");

  lcd.setCursor(0, 1);

  lcd.print("kWh:");

  lcd.print(kWh, 4);

  lcd.print("W");

  delay(2500);

  lastmillis = millis();

  Blynk.virtualWrite(V0, emon.Vrms);

  Blynk.virtualWrite(V1, emon.Irms);

  Blynk.virtualWrite(V2, emon.apparentPower);

  Blynk.virtualWrite(V3, kWh);

}

void setup()

{

  Serial.begin(9600);

  Blynk.begin(auth, ssid, pass);

  lcd.begin(16, 2);

  emon.voltage(35, vCalibration, 1.7); // Voltage: input pin, calibration, phase\_shift

  emon.current(34, currCalibration); // Current: input pin, calibration.

  timer.setInterval(5000L, myTimerEvent);

  lcd.setCursor(3, 0);

  lcd.print("IoT Energy");

  lcd.setCursor(5, 1);

  lcd.print("Meter");

  delay(3000);

  lcd.clear();

}

void loop()

{

  Blynk.run();

  timer.run();

}