```
#define BLYNK_PRINT Serial

#define BLYNK_TEMPLATE_ID "TMPL6AjdDpoO2"
#define BLYNK_TEMPLATE_NAME "IoT Solar Monitoring"
#define BLYNK_AUTH_TOKEN "IBMM7TM3LGJmHLnufrclc08dT8IiGvjR"

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

// EmonLibrary examples openenergymonitor.org, Licence GNU GPL V3
#include <Wire.h>
// #include <LiquidCrystal_I2C.h>
#include "DHT.h"
// LiquidCrystal_I2C lcd(0x27, 20, 4);

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


char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "student";
char pass[] = "iotstudent";

#define DHTPIN 25      // what digital pin we're connected to
#define DHTTYPE DHT22  // DHT 11
DHT dht(DHTPIN, DHTTYPE);
float t;
float h;

// battery volt
const int voltageSensor = 34;
float vOUT = 0.0;
float vIN = 0.0;
float R1 = 30000.0;
float R2 = 7100.0;
int vvalue = 0;
float charge = 0.00;



// solar volt
const int voltageSensor1 = 35;
float vOUT1 = 0.0;
```

```
float vsolar = 0.0;
float R11 = 30000.0;
float R21 = 7100.0;
int vvalue1 = 0;

// int ldr = 32;
// int ldrdata;



int relay = 26;
int iotdata;

int relay1 = 27;
int iotdata1;

// int servo = 19;
// int servodata;

int solarVoltage;

void setup() {
  Serial.begin(9600);
  // lcd.init();
  // lcd.backlight();
  // pinMode(ldr, INPUT);

  analogSetWidth(10);

  // pinMode(ldr, INPUT);

  pinMode(relay, OUTPUT);
  pinMode(relay1, OUTPUT);
  // pinMode(servo, OUTPUT);

  digitalWrite(relay, HIGH);
  digitalWrite(relay1, HIGH);

  dht.begin();


  Blynk.begin(auth, ssid, pass);
```

```
}

BLYNK_WRITE(V5)  // V0 is the number of Virtual Pin
{
  iotdata = param.asInt();
  Serial.println(iotdata);
  if (iotdata == 1) {
    digitalWrite(relay, LOW);
  }

  if (iotdata == 0) {
    digitalWrite(relay, HIGH);
  }
}

BLYNK_WRITE(V6)  // V0 is the number of Virtual Pin
{
  iotdata1 = param.asInt();
  Serial.println(iotdata1);
  if (iotdata1 == 1) {
    digitalWrite(relay1, LOW);
  }

  if (iotdata1 == 0) {
    digitalWrite(relay1, HIGH);
  }
}

// BLYNK_WRITE(V7)  // V0 is the number of Virtual Pin
// {
//   servodata = param.asInt();
//   Serial.println(servodata);
//   if (servodata == 1) {
//     digitalWrite(servo, HIGH);
//   }

//   if (servodata == 0) {
//     digitalWrite(servo, LOW);
//   }
// }


void loop() {
```

```
Blynk.run();


//Battery Volt Calculate
// vvalue = analogRead(voltageSensor);
// vOUT = (vvalue * 3.3) / 1024.0;
vOUT = analogReadMilliVolts(voltageSensor) / 1000.0;
vIN = vOUT / (R2 / (R1 + R2));


//Solar Volt Calculate
// vvalue1 = analogRead(voltageSensor1);
// vOUT1 = (vvalue1 * 3.3) / 1024.0;
vOUT1 = analogReadMilliVolts(voltageSensor1) / 1000.0;
vsolar = vOUT1 / (R21 / (R11 + R21));

// ldrdata = analogRead(ldr);
// ldrdata = map(ldrdata, 300, 4095, 0, 100);

// if (ldrdata <= 0) {
//   ldrdata = 0;
// }
// if (ldrdata >= 100) {
//   ldrdata = 100;
// }

//  svoltdata = analogRead(svolt);
// solarVoltage = analogReadMilliVolts(voltageSensor) * 0.00071 * (100 / 20);

//  charge = map(vIN, 10.00, 13.00, 0.00, 100.00);
charge = mapfloat(vIN, 7.10, 8.40, 0.00, 100.00);

if (charge <= 0.00) {
  charge = 0.00;
}
if (charge >= 100.00) {
  charge = 100.00;
}


t = dht.readTemperature();
h = dht.readHumidity();
```

```
  Serial.println(vsolar);
  // Serial.print("Volt:");
  // Serial.println(analogReadMilliVolts(voltageSensor));
  // Serial.println("");

  Blynk.virtualWrite(V0, t);
  Blynk.virtualWrite(V1, h);
  // Blynk.virtualWrite(V2, ldrdata);  //virtual pin V2
  Blynk.virtualWrite(V3, vsolar);   //virtual pin V2
  Blynk.virtualWrite(V4, charge);  //virtual pin V2

  delay(1000);
  // lcd.clear();
  // lcd.setCursor(0, 0);
  // lcd.print("Temp:");
  // lcd.print(int(t));
  // lcd.print("C");

  // lcd.setCursor(10, 0);
  // lcd.print("Hu:");
  // lcd.print(int(h));
  // lcd.print("%");

  // lcd.setCursor(0, 1);
  // lcd.print("Light Intensity:");
  // lcd.print(ldrdata);
  // lcd.print("%");

  // lcd.setCursor(0, 2);
  // lcd.print("Solar Volt: ");
  // lcd.print(vsolar);
  // lcd.print("V ");

  // lcd.setCursor(0, 3);
  // lcd.print("Bat Charge: ");
  // lcd.print(charge);
  // lcd.print("%");
}

float mapfloat(float x, float in_min, float in_max, float out_min, float out_max) {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```