**Internet Of Things- Solar Panel Monitoring System Using ESP8266 Nodemcu**

**ABSTRACT:**

This paper describes the monitoring of solar power by using internet of thing .The Internet of Things (IoT) refers to a system of interrelated, internet-connected objects that are able to collect and transfer data over a wireless network without human intervention. These IoT based technology is best suited for remote areas where solar Power plant are set up due to the ample availability of solar energy but regular access to the areas is very difficult and is not cost efficient .These IoT based technology are comprises of Solar Panel, NODE-MCU ESP8266,Voltage Sensor, Current Sensor, Temperature Senor etc

Solar power plants ought to be monitored for optimum power output. This helps retrieve economical power output from power plants whereas watching for faulty star panels, connections, dirt accumulated on panels lowering output and different such problems moving star performance. therefore, here we have a tendency to propose machine-driven an automatic IOT based mostly solar energy watching system that enables for automated solar energy watching from anyplace over the web. we have a tendency to use Arduino based mostly system to observe a 10Watt solar array parameters. Our system perpetually monitors the solar array and transmits the ability output to the IOT system over the web. Here we have a tendency to use IOT lizard to transmit solar energy parameters over the web to the IOT lizard server. It currently displays these parameters to the user exploitation a good graphical user interface and conjointly alerts users once the output falls below specific limits. This makes remotely watching of star plants terribly simple and guarantees the best power output.

The NODE-MCU ESP8266 controller used to monitor the parameters in solar panel. They monitor the Solar panel and transmit the output to the IOT Thingspeak transmits the solar power parameters in the Thingspeak server. The parameters Is displayed by using GUI and when the output falls below the specific limit it alerts the user, there is a problem in solar panel connections or any dust particles on the solar panel. This makes the monitoring of solar panel easier and ensure best power.

In this project we will monitor the solar panel using Blynk application through ESP8266 Nodemcu. The advantage of using the Nodemcu ESP8266 and Blynk application is, we can monitor the Solar Panel voltage and other parameters from anywhere in the world using our cell phone. So, this is basically an IoT based project.

**Components:**

DHT22 or DHT11 temperature sensor

NodeMCU

Voltage sensor

Relay module

Keyword: Solar Panel, IOT, Things speaker, Cloud Monitoring , NODE-MCU ESP8266

Code :

#include <DHT\_U.h>

#include <SimpleTimer.h>

//#include <BlynkSimpleEsp8266.h>

#define DHTPIN 2

#define DHTTYPE DHT11 // DHT 21 (AM2301)

DHT\_Unified dht(DHTPIN, DHTTYPE);

// inverter on voltage and cutoff voltage

float on\_voltage = 12.0;

float off\_voltage = 11.5;

// refresh interval

uint32\_t delayMS = 1000;

int pinValue1; // inverter virtual pin

int Vsensor = A0; // voltage sensor

int inverter = D2; // relay

// voltage sensor

float correctionfactor = 0; // adjust this for calibration

float vout = 0.0;

float vin = 0.0;

// two resistors 30k and 7.5k ohms

float R1 = 30000; //

float R2 = 7500; //

int value = 0;

char data[80];

SimpleTimer timer;

// Blynk token you received via email

char auth[] = "YOUR\_BLYNK\_AUTH\_TOKEN";

// Your WiFi credentials

char ssid[] = "Rabbani";

char pass[] = "Kousar@1357";

void setup() {

Serial.begin(9600);

pinMode(Vsensor, INPUT);

pinMode(inverter, OUTPUT);

digitalWrite(inverter, HIGH);

dht.begin();

Serial.println(F("Solar Power Monitor"));

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

//Blynk.virtualWrite(V5, 0);

sensor\_t sensor;

dht.temperature().getSensor(&sensor);

Serial.println(F("------------------------------------"));

Serial.println(F("Temperature Sensor"));

Serial.print (F("Sensor Type: ")); Serial.println(sensor.name);

Serial.print (F("Driver Ver: ")); Serial.println(sensor.version);

Serial.print (F("Unique ID: ")); Serial.println(sensor.sensor\_id);

Serial.print (F("Max Value: ")); Serial.print(sensor.max\_value); Serial.println(F("°C"));

Serial.print (F("Min Value: ")); Serial.print(sensor.min\_value); Serial.println(F("°C"));

Serial.print (F("Resolution: ")); Serial.print(sensor.resolution); Serial.println(F("°C"));

Serial.println(F("------------------------------------"));

// Print humidity sensor details.

dht.humidity().getSensor(&sensor);

Serial.println(F("Humidity Sensor"));

Serial.print (F("Sensor Type: ")); Serial.println(sensor.name);

Serial.print (F("Driver Ver: ")); Serial.println(sensor.version);

Serial.print (F("Unique ID: ")); Serial.println(sensor.sensor\_id);

Serial.print (F("Max Value: ")); Serial.print(sensor.max\_value); Serial.println(F("%"));

Serial.print (F("Min Value: ")); Serial.print(sensor.min\_value); Serial.println(F("%"));

Serial.print (F("Resolution: ")); Serial.print(sensor.resolution); Serial.println(F("%"));

Serial.println(F("------------------------------------"));

// Set delay between sensor readings

timer.setInterval(delayMS, timerEvent);

timerEvent();

}

void loop() {

// Blynk.run();

timer.run();

}

void timerEvent()

{

// Get temperature event and print its value.

sensors\_event\_t event;

// Virtual pin 1 (V1) has temperature value

dht.temperature().getEvent(&event);

if (isnan(event.temperature)) {

Serial.println(F("Error reading temperature!"));

//Blynk.virtualWrite(V1, "ERR");

}

else {

Serial.print(F("Temperature: "));

Serial.print(event.temperature);

Serial.println(F("°C"));

sprintf(data, "%.f", event.temperature);

//Blynk.virtualWrite(V1, data); // send data to blynk

}

// Get humidity event and print its value.

// Virtual pin 2 (V2) has humidity value

dht.humidity().getEvent(&event);

if (isnan(event.relative\_humidity)) {

Serial.println(F("Error reading humidity!"));

//Blynk.virtualWrite(V2, "ERR");

}

else {

Serial.print(F("Humidity: "));

Serial.print(event.relative\_humidity);

Serial.println(F("%"));

sprintf(data, "%.f", event.relative\_humidity);

//Blynk.virtualWrite(V2, data); // send data to blynk

}

float vtot = 0.0;

int loops = 10; // number of samples

// loop multiple times and get average reading

for (int i=0; i < loops; i++) {

vtot = vtot + analogRead(Vsensor);

}

value = vtot/loops;

// voltage calculation

vout = (value \* 3.3) / 1024.0; // 3.3V

vin = vout / (R2/(R1+R2));

vin = vin - correctionfactor;

Serial.print("Voltage: ");

Serial.print(vin, 4);

Serial.println("V");

// Virtual pin 0 (V0) has voltage value

sprintf(data, "%.1f", vin);

// Blynk.virtualWrite(V0, data); // send voltage value to blynk

if (vin < off\_voltage && digitalRead(inverter) == LOW) {

digitalWrite(inverter, HIGH);

Serial.println("TURNING RELAY OFF");

}

if (vin > on\_voltage && digitalRead(inverter) == HIGH) {

digitalWrite(inverter, LOW);

Serial.println("TURNING RELAY ON");

}

if (digitalRead(inverter) == LOW) {

// Blynk.virtualWrite(V5, 255);

Serial.println("Inverter is ON");

} else {

// Blynk.virtualWrite(V5, 0);

Serial.println("Inverter is OFF");

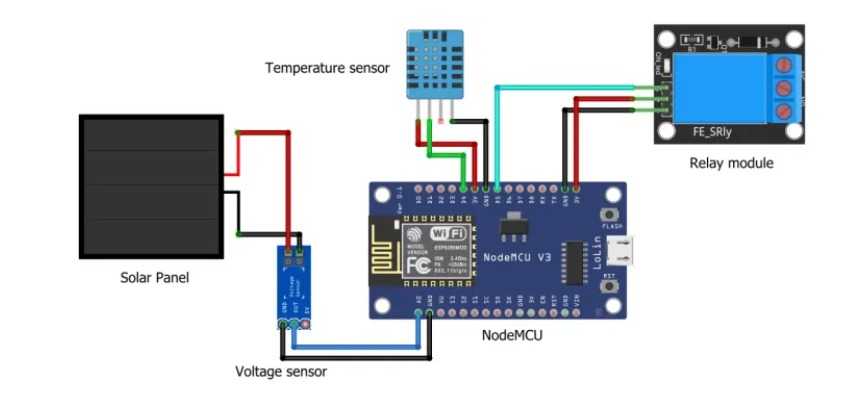
}

Serial.println("------------------");

Serial.println("");

}

**CIRCUIR DIAGRAM :**

****

OUTPUT:-

