#include <SoftwareSerial.h>

#include <Servo.h>

#include <LiquidCrystal.h>

#define RX 10

#define TX 12

#define ESP\_CH\_EN 8

#define SolarIin A2

#define SolarVin A3

String AP = "Symcore"; // ACCESS POINT NAME

String PASS = "SymcoreWifi@123"; // AP PASSWORD

String API = "A7HGCVKDDVJRX8Q3"; // THINGSPEAK API KEY

String HOST = "api.thingspeak.com";

String PORT = "80";

String Vfield = "field1"; //THINGSPEAK VOLTAGE FIELD

String Ifield = "field2"; //THINGSPEAK CURRENT FIELD

String Pfield = "field3"; //THINGSPEAK POWER FIELD

int countTrueCommand;

int countTimeCommand;

int initial\_position = 90;

int LDR1 = A1; //connect The LDR1 on Pin A0

int LDR2 = A0; //Connect The LDR2 on pin A1

int error = 75;

int servopin = 9;

const int rs = 6, en = 7, d4 = 2, d5 = 3, d6 = 4, d7 = 5;

double Voltage = 0;

double Current = 0;

double Power = 0;

double valVoltage = 0;

double valCurrent = 0;

double valPower = 0;

boolean found = false;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

SoftwareSerial ESP8266(RX, TX);

Servo SG90;

void setup()

{

//solar tracker setup start

SG90.attach(servopin);

pinMode(LDR1, INPUT);

pinMode(LDR2, INPUT);

SG90.write(initial\_position);

delay(2000);

//solar tracker setup end

//Solar power Monitoring Setup Start

pinMode(SolarVin, INPUT);

pinMode(SolarIin, INPUT);

pinMode(ESP\_CH\_EN, OUTPUT);

lcd.begin(16, 2);

Serial.begin(9600);

ESP8266.begin(115200);

digitalWrite(ESP\_CH\_EN, HIGH);

sendCommand("AT", 5, "OK");

sendCommand("AT+CWMODE=1", 5, "OK");

sendCommand("AT+CWJAP=\"" + AP + "\",\"" + PASS + "\"", 20, "OK");

digitalWrite(ESP\_CH\_EN, LOW);

lcd.setCursor(0, 0);

lcd.print("Solar Tracking &");

lcd.setCursor(0, 1);

lcd.print("Power Monitoring &");

//Solar power Monitoring Setup END

}

void loop() {

int starttime = millis();

int endtime = starttime;

while ((endtime - starttime) <= 12000) // do this loop for up to 10000mS

{

SolarTracking();

endtime = millis();

}

valVoltage = getVoltageData();

valCurrent = getCurrentData();

valPower = getPowerData();

String LCDdataL1 = "V =" + String(valVoltage) + " I =" + String(valCurrent);

String LCDdataL2 = "P =" + String(valPower);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(LCDdataL1);

lcd.setCursor(0, 1);

lcd.print(LCDdataL2);

digitalWrite(ESP\_CH\_EN, HIGH);

String getData = "GET /update?api\_key=" + API + "&" + Vfield + "=" + String(valVoltage) + "&" + Ifield + "=" + String(valCurrent) + "&" + Pfield + "=" + String(valPower);

sendCommand("AT+CIPMUX=1", 5, "OK");

sendCommand("AT+CIPSTART=0,\"TCP\",\"" + HOST + "\"," + PORT, 15, "OK");

sendCommand("AT+CIPSEND=0," + String(getData.length() + 4), 4, ">"); //send data to thingspeak

ESP8266.println(getData); //print on serial monitor

delay(20);

countTrueCommand++;

sendCommand("AT+CIPCLOSE=0", 5, "OK");

digitalWrite(ESP\_CH\_EN, LOW);

}

int SolarTracking()

{

int R1 = analogRead(LDR1); // read LDR 1

int R2 = analogRead(LDR2); // read LDR 2

int diff1 = abs(R1 - R2);

int diff2 = abs(R2 - R1);

if ((diff1 <= error) || (diff2 <= error)) {

}

else {

if ((R1 > R2) && (SG90.read() >= 35))

{

initial\_position = --initial\_position;

}

if ((R1 < R2) && (SG90.read() <= 120))

{

initial\_position = ++initial\_position;

}

}

SG90.write(initial\_position);

delay(35);

}

double getVoltageData()

{

Voltage = ((0.004883 \* analogRead(SolarVin)) \* 52.7) / 37.7;

return Voltage; // Replace with

}

double getCurrentData()

{

Current = ((0.004883 \* analogRead(SolarIin)) \* 1000) / 100; //measure current in milliamps

return Current; // Replace with

}

double getPowerData()

{

valVoltage = getVoltageData();

valCurrent = getCurrentData();

Power = valVoltage \* valCurrent;

return Power; // Replace with

}

void sendCommand(String command, int maxTime, char readReplay[])

{

Serial.print(countTrueCommand);

Serial.print(". at command => ");

Serial.print(command);

Serial.print(" ");

while (countTimeCommand < (maxTime \* 1))

{

ESP8266.println(command);//at+cipsend

if (ESP8266.find(readReplay)) //ok

{

found = true;

break;

}

countTimeCommand++;

}

if (found == true)

{

Serial.println("OYI");

countTrueCommand++;

countTimeCommand = 0;

}

if (found == false)

{

Serial.println("Fail");

countTrueCommand = 0;

countTimeCommand = 0;

}

found = false;

}