1. **Chức năng của các thư viện sử dụng trong chương trình điều khiển:**

* Thư viện giao tiếp với LCD theo chuẩn I2C:

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

* Kết nối WiFi cho Node MCU 1.0:

#include <ESP8266WiFi.h>

#include <WiFiSupportESP8266.h>

* Kết nối Node MCU 1.0 và Blynk Server:

#include <BlynkSimpleEsp8266.h>

* Điều khiển chuông báo động:

#include <ControlDigitalOutput.h>

* Giao tiếp với cảm biến dòng điện và điện áp PZEM004T:

#include <PZEM004T.h>

1. **Một số khai báo chân cho kết nối phần cứng và các hằng số điều khiển:**

#define PIN\_RELAY\_LOAD1 0 // D3

#define PIN\_RELAY\_LOAD2 2 // D4

#define PIN\_RELAY\_LOAD3 14 // D5

#define PIN\_BUZZER 12// D6

#define ACTIVE\_BUZZER LOW

#define BLYNK\_PRINT Serial

#define BAUD\_RATE 115200

#define TIMEOUT 60000

#define TIME\_UPDATE\_DATA 2000L

#define debug\_port Serial

1. **Khai báo giao điện điều khiển và kết nối đến Blynk Server**

#define PIN\_GAUGE\_CURRENT V0 // Nút hiển thị dòng điện trên giao diện Blynk

#define PIN\_GAUGE\_VOLTAGE V3 // Nút hiển thị điện áp trên giao diện Blynk

#define PIN\_CURRENT\_SETUP V1 // Nút cài đặt dòng điện định mức trên giao diện Blynk

#define PIN\_CURRENT\_ALRAM V2 // Nút điều khiển cảnh báo quá dòng trên giao diện Blynk

char auth[] = "3319436c3aa0422a99a14802dd51fef7"; // Mã chứng thực để kết nối Blynk

1. **Sử dụng thư viện các thư viện ở phần trên để tạo ra các đối tượng giao tiếp:**

* Tạo đối tượng giao tiếp với cảm biến dòng điện và điện áp PZEM004T:

PZEM004T pzem(&Serial);

IPAddress ip(192,168,1,1);

* Giao tiếp với LCD với chuẩn I2C (địa chỉ I2C là 0x27, 16 cột, 2 dòng):

LiquidCrystal\_I2C lcd(0x27, 16, 2);

* Kết nối WiFi cho thiết bị:

WiFiSupportESP8266 wifiSupport(LED\_BUILTIN, LOW);

(Chưa kết nối được WiFi thì LED\_BUILTIN sẽ sáng lên, kết nối được WiFi thì LED\_BUILTIN sẽ tắt).

* Điều khiển chuông báo động:

ControlDigitalOutput buzzer(PIN\_BUZZER, ACTIVE\_BUZZER);

* Gửi dữ liệu lên Server Blynk sau thời gian định thời:

BlynkTimer timer;

1. **Chức năng của các hàm (chương trình con) trong chương trình chính:**
   * Kết nối với LCD:

lcd.begin();

lcd.backlight();

lcd.clear();

* Kết nối với cảm biến dòng điện và điện áp:

lcd.print("Init PZEM");

pzem.setAddress(ip);

* Kết nối WiFi cho Node MCU:

if (!wifiSupport.isSmartConfig(TIMEOUT)) {

ESP.restart();

}

* Kết nối với Server Blynk và tự động cập nhật dữ liệu sau thời gian định thời:

Blynk.config(auth);

timer.setInterval(TIME\_UPDATE\_DATA, updatePowerMetter);

* Giữ kết nối với Server Blynk và liên tự cập nhật dữ liệu lên Server:

Blynk.run();

timer.run();

* Đọc giá trị từ cảm biến dòng điện và điện áp PZEM004T: PowerMetter getPowerMetter() trả về giá trị có kiểu: PowerMetter

struct PowerMetter {

float voltage;

float current;

float power;

float energy;

};

PowerMetter getPowerMetter() {

PowerMetter powerMetter;

powerMetter.voltage = pzem.voltage(ip);

if (powerMetter.voltage < 0.0) {

powerMetter.voltage = 0.0;

}

powerMetter.current = pzem.current(ip);

if (powerMetter.current < 0.0) {

powerMetter.current = 0.0;

}

powerMetter.power = pzem.power(ip);

if (powerMetter.power < 0.0) {

powerMetter.power = 0.0;

}

powerMetter.energy = pzem.energy(ip);

if (powerMetter.energy < 0.0) {

powerMetter.energy = 0.0;

}

return powerMetter;

}

* Hiển thị giá trị dòng điện và điện áp lên LCD với hàm: void displayLCD(PowerMetter powerMetter)

void displayLCD(PowerMetter powerMetter) {

String line1 = "Dien ap:" + String(powerMetter.voltage) + "V";

String line2 = "Dong dien:" + String(powerMetter.current) + "A";

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(line1);

lcd.setCursor(0, 1);

lcd.print(line2);

}

* Gửi dữ liệu dòng điện và điện áp lên Server Blynk với hàm: void displayGaugeVirtual(PowerMetter powerMetter)

void displayGaugeVirtual(PowerMetter powerMetter) {

Blynk.virtualWrite(PIN\_GAUGE\_CURRENT, powerMetter.current);

Blynk.virtualWrite(PIN\_GAUGE\_VOLTAGE, powerMetter.voltage);

}

* Đọc giá trị đặt của dòng điện định mức được cài đặt từ Server Blynk (thông qua cài đặt trên App Blynk):

BLYNK\_WRITE(PIN\_CURRENT\_SETUP) {

float current = param.asFloat();

// debug\_port.println(current);

setMaxCurrent(current);

}

* Đọc giá trị điều khiển chuông cảnh báo từ Server Blynk:

BLYNK\_WRITE(PIN\_CURRENT\_ALRAM) {

bool isAlarm = param.asInt();

if (isAlarm) {

buzzer.on();

} else {

buzzer.off();

}

}

* Đồng bộ trạng thái từ Server Blynk:

BLYNK\_CONNECTED() {

Blynk.syncAll();

}

* Lấy, cài đặt và kiểm tra giá trị của dòng điện định mức: maxCurrent

void setMaxCurrent(float \_current) {

maxCurrent = \_current;

}

float getMaxCurrent() {

return maxCurrent;

}

bool checkCurrentValid(float \_current) {

return \_current <= getMaxCurrent();

}

* Đọc giá trị từ cảm biến dòng điện và điện áp PZEM004T, kiểm tra điều kiện cảnh báo, hiển thị lên LCD và gửi lên Server Blynk với hàm void updatePowerMetter() (kết hợp các hàm được trình bày ở phía trên):

void updatePowerMetter() {

PowerMetter powerMetter = getPowerMetter();

if (!checkCurrentValid(powerMetter.current)) {

buzzer.on();

updateBuzzerAlarm(true);

} else {

buzzer.off();

updateBuzzerAlarm(false);

}

displayGaugeVirtual(powerMetter);

displayLCD(powerMetter);

}