**Chương 1**

**TỔNG QUAN**

* 1. Lý do chọn đề tài

Hiện nay, ngành kinh tế và công nghiệp nước ta đang phát triển mạnh mẽ. Nhu cầu sản xuất và tiêu thụ điện năng trong lĩnh vực nông nghiệp, công nghiệp và sinh hoạt gia tăng nhanh chóng. Trong đó, công nghiệp luôn là ngành tiêu thụ điện năng lớn nhất. Tuy nhiên, chất lượng điện ở Việt Nam vẫn còn không ổn định. Điều này gây ảnh hưởng rất lớn đến quá trình sản xuất và hoạt động của các nhà máy, xí nghiệp như gây ảnh hưởng đến chất lượng sản phẩm, hàng hóa, gây gián đoạn quy trình làm việc, gây hư hỏng thiết bị… Do đó, điện năng ổn định thực sự đóng góp một phần quan trọng đến doanh thu của các nhà máy, xí nghiệp. Việc thiết kế hệ thống giám sát điện năng để quản lý chất lượng điện và sự hoạt động ổn định của tải là vô cùng cần thiết. Xuất phát từ tính tất yếu trên, em đã chọn đề tài **“Thiết kế và thi công hệ thống đo và giám sát điện năng”** để qua đó, từ các thông số đo được từ mạng điện sẽ giúp chúng ta đưa ra được các phương án cải thiện chất lượng điện cũng như không làm ảnh hưởng đến quy trình sản xuất của nhà máy, xí nghiệp.

* 1. Phạm vi nghiên cứu

Thiết kế hệ thống giám sát điện năng và phân tích các thông số đo được từ mạng điện dựa vào vi điều khiển, các linh kiện và các module cần thiết có sẵn trên thị trường.

* 1. Bố cục của đồ án

Để tìm hiểu đề tài này, báo cáo được chia thành 4 chương.

Chương 1: Tổng quan

Chương 2: Cơ sở lý thuyết

Chương 3: Thiết kế, thi công mạch

Chương 4: Kết luận và hướng phát triển

**Chương 2**

**CƠ SỞ LÝ THUYẾT**

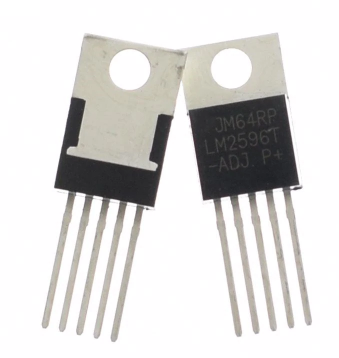
**2.1.** IC LM2596

### **2.1.1.** Giới thiệu

IC LM2596 là một IC ổn áp dạng xung DC-DC, là một mạch tích hợp với chức năng điều chỉnh làm giảm điện áp, có khả năng lái tải với dòng ra lên đến 3A, có chu kì đóng/ngắt lên đến 150 kHz cho hiệu suất làm việc cao.

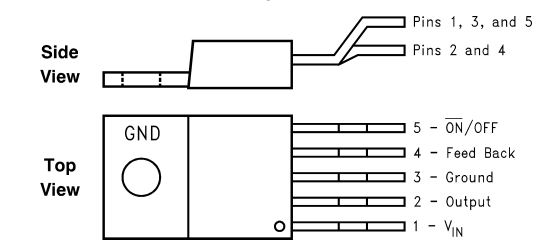
Đặc điểm:

* Điều chỉnh điện áp đầu ra phạm vi 1.23 V – 37 V.
* Đảm bảo dòng ngõ ra 3 A.
* Phạm vi điện áp đầu vào rộng lên đến 40 V.
* Chế độ chờ công suất thấp.
* Bộ dao động nội tần số cố định 150 kHz.
* Ngưng hoạt động khi quá nhiệt và có quá dòng.



**Hình 2.1:** IC LM2596

### **2.1.2.** Sơ đồ chân



**Hình 2.2:** Sơ đồ chân IC LM2596

IC LM2596 có 5 chân:

*Chân 1: Vin*

*Chân 2: Output*

*Chân 3: Ground*

*Chân 4: Feedback*

*Chân 5: On/Off*

**Bảng 2.1:** Chức năng các chân IC LM2596:

|  |  |
| --- | --- |
| Vin | Chân này là nguồn cung cấp đầu vào LM2596. Để giảm thiểu điện áp tạm thời và để cung cấp các dòng cần thiết, chúng ta phải gắn ở đầu vào các tụ bypass. |
| Output | Chân ngõ ra của mạch.Điện áp bão hòa Vsat của đầu ra này thường là 1,5V. |
| Ground | Chân nối đất. |
| Feedback | Chân này cảm nhận điện áp đầu ra được điều chỉnh để phản hồi. |
| ON/OFF | Chân cho phép mạch điều chỉnh chuyển mạch ngưng hoạt động bằng cách sử dụng mức tín hiệu logic, do đó giảm tổng nguồn cung đầu vào xuống khoảng 80uA. Điện áp ngưỡng thường là 1.3V. |

### **2.1.3.** Sơ đồ nguyên lý

**Hình 2.3:** Sơ đồ nguyên lý IC LM2596

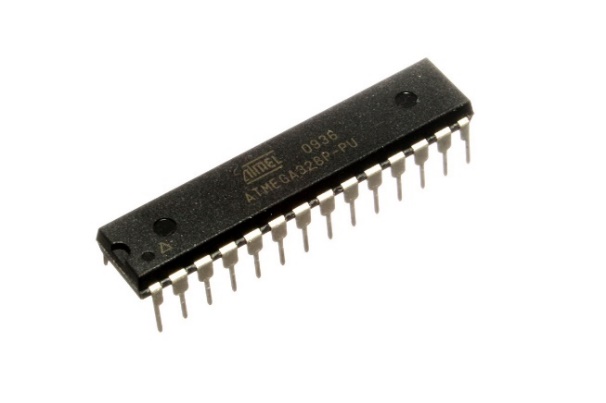
**Bảng 2.2:** Thông số kỹ thuật của IC LM2596

|  |  |
| --- | --- |
| Điện áp vào nhỏ nhất | 4.5V |
| Điện áp vào lớn nhất | 40V |
| Điện áp ra nhỏ nhất | 3.3V |
| Điện áp ra lớn nhất | 37V |
| Dòng điện ngõ ra lớn nhất | 3A |
| Tần số chuyển mạch nhỏ nhất | 110kHz |
| Tần số chuyển mạch lớn nhất | 173khz |
| Nhiệt độ hoạt động | -40°C ~ 125°C |

**2.2.** IC ATmega 328P

### **2.2.1**. Giới thiệu

ATmega 328 có tên đầy đủ là ATmega 328P-PU. ATmega 328 là một chíp vi điều khiển được sản xuất bời hãng Atmel thuộc họ MegaAVR có sức mạnh hơn hẳn ATmega8. ATmega 328 là một bộ vi điều khiển 8 bít dựa trên kiến trúc RISC bộ nhớ chương trình 32kB ISP flash có thể ghi xóa hàng nghìn lần, 1kB EEPROM, một bộ nhớ RAM vô cùng lớn trong thế giới vi xử lý 8 bít (2kB SRAM).

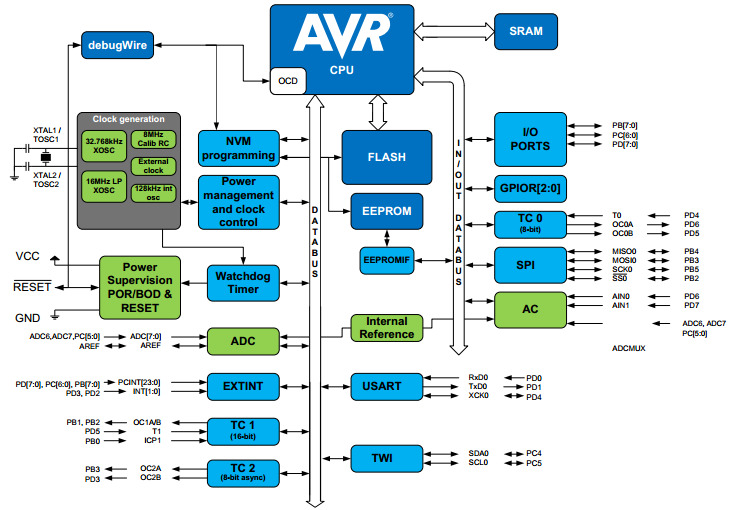
Những tính năng tuyệt vời của nó bao gồm hiệu quả chi phí, tiêu thụ năng lượng thấp, lập trình cho mục đích an ninh, bộ đếm thời gian thực với bộ dao động riêng biệt.... Nó thường được sử dụng trong các ứng dụng hệ thống nhúng.

**Hình 2.4:** IC ATmega 328

**Bảng 2.3:** Bảng tóm tắt cấu hình IC ATmega 328

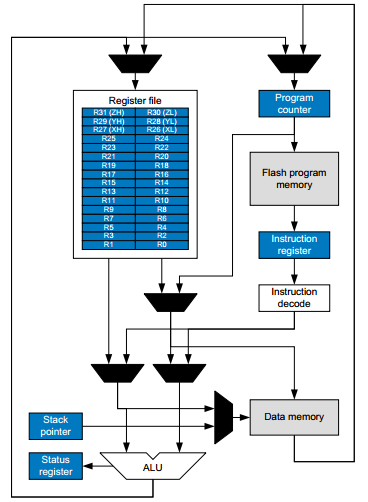
|  |  |
| --- | --- |
| Số chân | 28 |
| Bộ nhớ chương trình (Flash) | 32kB |
| Bộ nhớ SRAM | 2kB |
| Bộ nhớ EEPROM | 1kB |
| Số chân I/O | 23 |
| Số kênh ADC 10 bit | 8 kênh |
| Số timer/counter | 3 |

### **2.2.2**. Sơ đồ khối

Sơ đồ khối hiển thị các mạch bên trong và luồng của chương trình của bất kỳ vi điều khiển nào. Sơ đồ khối ATmega 328 được thể hiện trong hình bên dưới:

**Hình 2.5:** Sơ đồ khối IC ATmega 328

### **2.2.3.** Lõi CPU của ATmega 328

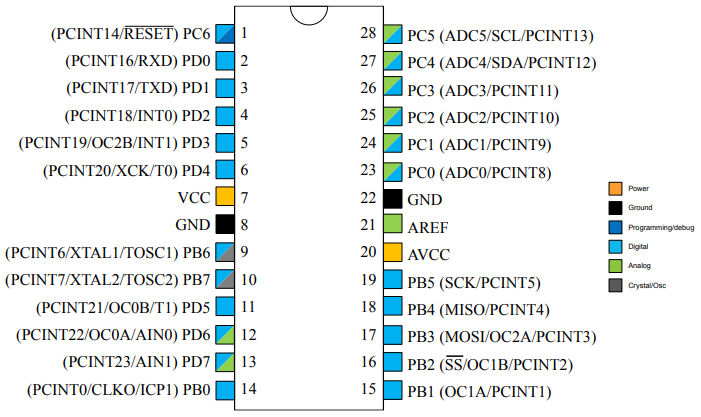
Chức năng chính của lõi CPU là đảm bảo thực hiện chương trình chính xác. Do đó, CPU phải có khả năng truy cập bộ nhớ, thực hiện tính toán, điều khiển các thiết bị ngoại vi và xử lý các ngắt.

**Hình 2.6:** Sơ đồ khối của kiến trúc ATmega 328

Để có được hiệu năng cao nhất và khả năng làm việc song song , ATmega 328 sử dụng cấu trúc Harvard – với sự phân chia bộ nhớ và các bus cho chương trình và dữ liệu . Các lệnh trong bộ nhớ chương trình thì được thực thi với 1 cấp sử lí liên lệnh đơn .Trong khi lệnh được đang được xử lí thì lệnh tiếp theo được nạp tiếp từ bộ nhớ chương trình . Khái niệm này kích hoạt lệnh để thực thi trong mỗi chu kì xung nhịp clock. Bộ nhớ chương trình là bộ nhớ flash có thể lập trình lại được ở trong hệ thống.

### **2.2.4.** Sơ đồ chân

Thông qua sơ đồ chân, chúng ta có thể hiểu được cấu hình của các chân của các dòng vi điều khiển.



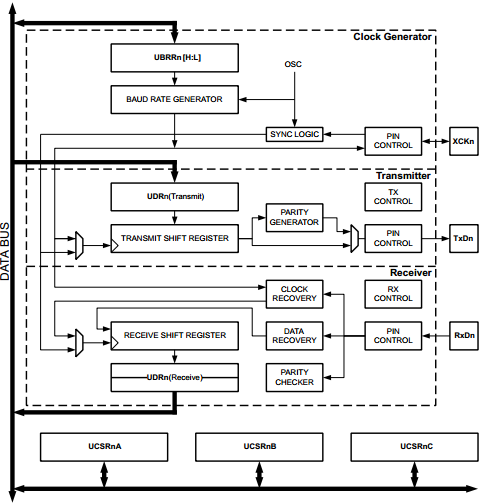
**Hình 2.7:** Sơ đồ chân IC ATmega 328

**Bảng 2.4:** Mô tả các chân IC ATmega 328

|  |  |
| --- | --- |
| VCC | Nguồn cung cấp |
| GND | Chân nối đất |
| PB[5:0]  PC[5:0]  PD[7:0] | Port B, C, D là port I/O với các điện trở kéo lên bên trong. Bộ đệm đầu ra của các port này có các đặc tính ngõ ra với cả khả năng lưu trữ dòng sink và source cao. Nếu là ngõ vào, các chân của các port này sẽ được điện trở kéo xuống bên ngoài, sẽ là dòng source nếu các điện trở kéo lên được kích hoạt. |
| PB[7:6] | Hai chân kết nối thạch anh tạo dao động. |
| PC6/RESET | Chân Reset lại dữ liệu của vi điều khiển |
| AVCC | AVCC là chân cung cấp điện áp cho bộ chuyển đổi A/D (ADC). Nó nên được kết nối với VCC ngay cả khi ADC không sử dụng. |
| AREF | AREF là chân điện áp tham chiếu tương tự cho bộ chuyển đổi A/D. |

### **2.2.5.** UART

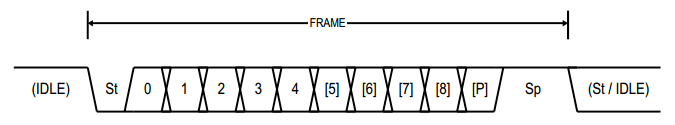
UART (Universal Asynchronous Receive/Transmit) là chuẩn giao tiếp truyền nhận dữ liệu không đồng bộ. Các đặc điểm của chuẩn UART trên ATmega 328:

* Hoạt động song công hoàn toàn.
* Hoạt động không đồng bộ hoặc đồng bộ.
* Trình tạo tốc độ baud độ phân giải cao.
* Hỗ trợ khung truyền nối tiếp với 5, 6, 7, 8 hoặc 9 bit dữ liệu và 1 hoặc 2 bit stop.
* Phát hiện tràn dữ liệu hoặc lỗi khung truyền.
* Lọc nhiễu.

**Hình 2.8:** Sơ đồ khối UART của ATmega 328

Các thành phần chính của UART gồm bộ phát xung nhịp, bộ truyền và bộ nhận dữ liệu. Các thanh ghi điều khiển được chia sẻ bởi tất cả các bộ phận. Khối logic phát xung clock bao gồm khối logic đồng bộ các xung clock bên ngoài và máy phát tốc độ baud. Chân XCK (xung truyền dữ liệu) thì chỉ được sử dụng bởi chế độ truyền đồng bộ. Bộ phát dữ liệu bao gồm một bộ đệm ghi đơn, một thanh ghi dịch nối tiếp, bộ tạo chẵn lẻ và khối điều khiển logic cho việc xử lý các định dạng khung nối tiếp khác nhau. Bộ đệm ghi cho phép truyền dữ liệu liên tục mà không có bất cứ độ trễ nào giữa các khung. Bộ thu dữ liệu là bộ phận phức tạp nhất của module UART vì xung clock của nó và các bộ phận phục hồi dữ liệu. Các bộ phận khôi phục dữ liệu được sử dụng cho việc để nhận dữ liệu không đồng bộ. Thêm vào đó để khôi phục các bộ phận này, bộ thu bao gồm bộ kiểm tra chẵn lẻ, khối điều khiển logic, một thanh ghi dịch và bộ đệm thu nhận dữ liệu 2 cấp (UDR).

* ***Định dạng khung truyền:***

Một khung truyền bắt đầu với bit start, tiếp theo là các bit dữ liệu (từ 5 đến 9 bit): đầu tiên là bit dữ liệu có trọng số thấp nhất, sau đó là các bit dữ liệu tiếp theo và kết thúc là bit có trọng số cao nhất. Nếu được kích hoạt, bit parity được chèn sau các bit dữ liệu, trước 1 hoặc 2 bit stop. Khi 1 khung truyền hoàn chỉnh được truyền đi, có thể được theo dõi trực tiếp bởi một khung mới hoặc đường truyền có thể được đặt ở trạng thái không hoạt động (mức cao).

**Hình 2.9:** Sơ đồ định dạng khung truyền UART của ATmega 328

Trong đó:

* + St: bit start, luôn ở mức thấp.
* (n): các bit dữ liệu (từ 0 đến 8 ).
* P: bit chẵn lẻ, có thể là chẵn hoặc lẻ.
* Sp: bit stop, luôn ở mức cao
* IDLE : không có quá trình chuyển phát nào trên đường giao tiếp dữ liệu. Một đường IDLE phải ở mức cao

## **2.3.** Module đo điện năng PZEM004T

### **2.3.1.** Giới thiệu

Mạch đo điện AC giao tiếp UART PZEM004T được sử dụng để đo và theo dõi gần như hoàn toàn các thông số về điện năng AC của mạch điện như điện áp hoạt động, dòng tiêu thụ, công suất và năng lượng tiêu thụ của các thiết bị điện.



**Hình 2.10:** Module PZEM004T

Chức năng của module:

* Chức năng đo các tham số điện năng.
* Chức năng báo động quá tải.
* Chức năng cài đặt ngưỡng báo động nguồn.
* Chức năng cài đặt lại của phím năng lượng.
* Lưu trữ dữ liệu khi tắt nguồn.
* Chức năng hiển thị kỹ thuật.
* Chức năng giao tiếp Serial.

**Bảng 2.5:** Thông số kỹ thuật của module PZEM004T

|  |  |
| --- | --- |
| Điện áp làm việc | 80 ~ 260VAC |
| Điện áp kiểm tra | 80 ~ 260VAC |
| Công suất định mức | 100A/22000W |
| Tần số hoạt động | 45 ~ 65Hz |
| Độ chính xác đo lường | 1 cấp |

### **2.3.2.** Sơ đồ đi dây

**Hình 2.11:** Sơ đồ đi dây của module PZEM004T

Hệ thống đi dây của module này được chia thành hai phần:

* + Phần kiểm tra điện áp và dòng điện đầu vào.
  + Phần kết nối giao diện truyền dữ liệu nối tiếp TTL.

Giới hạn đo của module:

* Công suất: Phạm vi kiểm tra: 0 ~ 22kW.
* Năng lượng: Phạm vi kiểm tra: 0 ~ 9999kWh.
* Điện áp: Phạm vi kiểm tra: 80 ~ 260VAC..
* Dòng điện: Phạm vi kiểm tra: 0 ~ 100A.

***Phương pháp đặt lại dữ liệu:*** Trên module có duy nhất một nút nhấn dùng để xóa dữ liệu đã lưu. Nhấn và giữ phím trong 5 giây, sau đó nhả phím. Nhấn nhanh phím lần nữa, sau đó dữ liệu năng lượng sẽ bị xóa, bây giờ hoạt động đặt lại được hoàn thành. Nếu nhấn lâu trong 5 giây nữa có nghĩa là thoát trạng thái đặt lại.

### **2.3.3.** Các lệnh giao tiếp

***Cấu trúc lệnh***

|  |  |  |
| --- | --- | --- |
| Header | Data 1 – Data 5 | Checksum |
| 1 byte | 5 byte | 1 byte |

* ***Lệnh lấy giá trị điện áp***

|  |  |  |
| --- | --- | --- |
| Header | Data 1 – Data 5 | Checksum |
| B0 | C0 A8 01 01 00 | 1A |

***Phản hồi từ PZEM-004T***

|  |  |  |
| --- | --- | --- |
| Header | Data 1 – Data 5 | Checksum |
| A0 | 00 E6 02 00 00 | 88 |

Dữ liệu điện áp trả về là D1 D2 D3 = 00 E6 02, 00 E6 đại diện cho số nguyên của điện áp, 02 đại diện cho số thập phân của điện áp, số thập phân là một chữ số.

Chuyển đổi 00 E6 thành thập phân là 230, chuyển đổi 02 thành thập phân là 2, vì vậy giá trị điện áp hiện tại là 230.2V

* ***Lệnh lấy giá trị dòng điện***

|  |  |  |
| --- | --- | --- |
| Header | Data 1 – Data 5 | Checksum |
| B1 | C0 A8 01 01 00 | 1B |

***Phản hồi từ PZEM004T***

|  |  |  |
| --- | --- | --- |
| Header | Data 1 – Data 5 | Checksum |
| A1 | 00 11 20 00 00 | D2 |

Dữ liệu dòng điện trả về là D2 D3 = 11 20,11 đại diện cho số nguyên của dòng điện, 20 đại diện cho số thập phân của dòng điện, số thập phân dòng điện là hai chữ số.

Chuyển đổi 11 thành thập phân là 17, chuyển đổi 20 thành số thập phân là 32, do đó giá trị dòng điện hiện tại là 17,32 A.

* ***Lệnh lấy giá trị công suất***

|  |  |  |
| --- | --- | --- |
| Header | Data 1 – Data 5 | Checksum |
| B2 | C0 A8 01 01 00 | 1C |

***Phản hồi từ PZEM004T***

|  |  |  |
| --- | --- | --- |
| Header | Data 1 – Data 5 | Checksum |
| A2 | 08 98 00 00 00 | 42 |

Dữ liệu công suất trả về là D1 D2 = 08 98, chuyển đổi 08 98 thành thập phân là 2200, do đó giá trị công suất hiện tại là 2200W.

* ***Lệnh lấy giá trị năng lượng***

|  |  |  |
| --- | --- | --- |
| Header | Data 1 – Data 5 | Checksum |
| B3 | C0 A8 01 01 00 | 1D |

***Phản hồi từ PZEM004T***

|  |  |  |
| --- | --- | --- |
| Header | Data 1 – Data 5 | Checksum |
| A3 | 01 86 9F 00 00 | C9 |

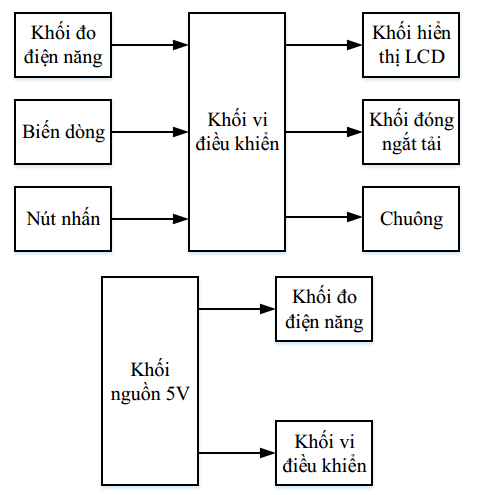
Dữ liệu năng lượng trả về là D1 D2 D3 = 01 86 9F, chuyển đổi 01 86 9F sang thập phân là 99999, vì vậy năng lượng tích lũy là 99999Wh.

**Chương 3**

**THIẾT KẾ, THI CÔNG MẠCH**

**3.1.** Sơ đồ khối của hệ thống

**3.1.1.** Sơ đồ khối



**Hình 3.1:** Sơ đồ khối của mạch đo và giám sát điện năng

**3.1.2.** Chức năng của từng khối

Khối nguồn 5V: Cung cấp nguồn nuôi linh kiện trong mạch.

Khối đo điện năng: Đo điện áp, công suất, điện năng tiêu thụ.

Biến dòng: Đo dòng điện.

Nút nhấn: Chọn chế độ đo, đặt trước điện áp ngưỡng .

Khối vi điều khiển: Chuyển đổi giá trị đại lượng nhận được từ số hex thành số thập phân và xử lý khối hiển thị.

Khối hiển thị LCD: Hiển thị, quan sát các thông số.

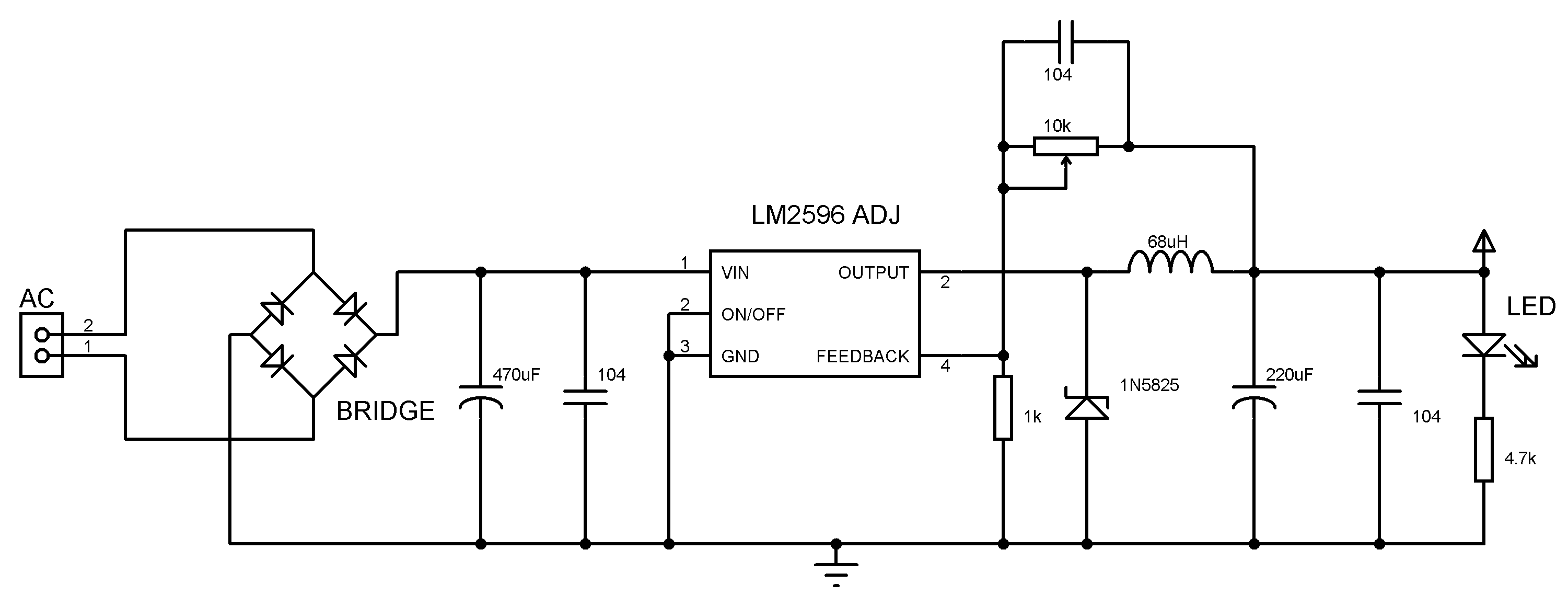
Khối đóng ngắt tải: Đóng ngắt tải khi điện áp vượt điện áp ngưỡng.

Chuông: Báo động khi điện áp vượt điện áp ngưỡng.

## **3.2.** Thiết kế hệ thống

**3.2.1.** Khối nguồn

a. Sơ đồ nguyên lý



**Hình 3.2:** Sơ đồ nguyên lý của khối nguồn

Ở đây ta cần điện áp thay đổi tối đa là 15V nên sử dụng thêm biến trở. Giá trị của biến trở áp dụng theo công thức:

= 11 kΩ

Do đó ta chọn biến trở 10 kΩ

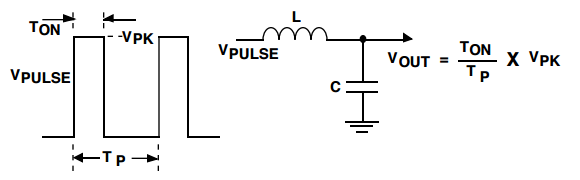
Tính toán điện trở cho led báo hiệu:

Ta chọn điện trở 4.7 kΩ để không bị chói mắt.

Các tham số còn lại thao khảo datasheet, phụ lục trang 44.

b. Nguyên lý hoạt động:

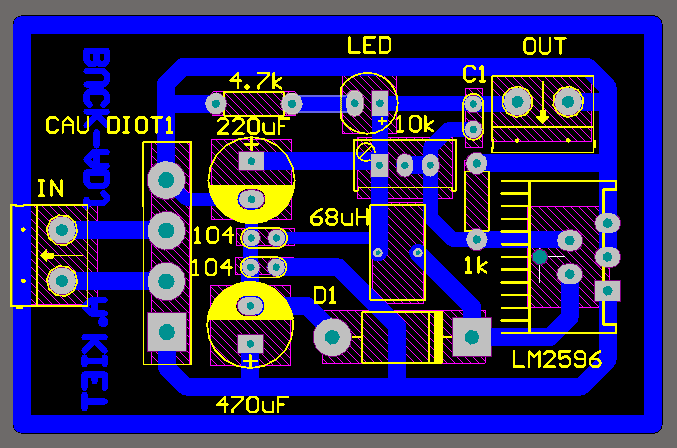
Nguồn điện xoay chiều 220V đi qua cầu diode và IC nguồn xung LM2596 trở thành dòng điện một chiều có dạng xung vuông với tần số xác định nào đó, dùng để điều chỉnh độ rộng (PWM, hay thường được gọi là băm xung).

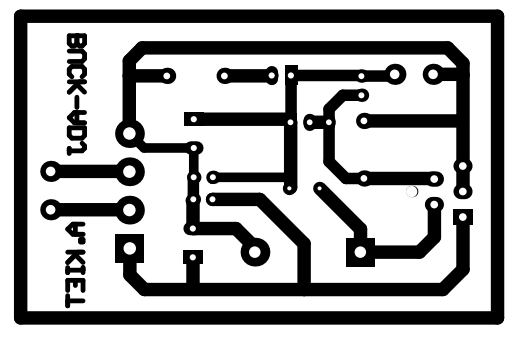


**Hình 3.3:** Sơ đồ dạng sóng ngõ ra của khối nguồn

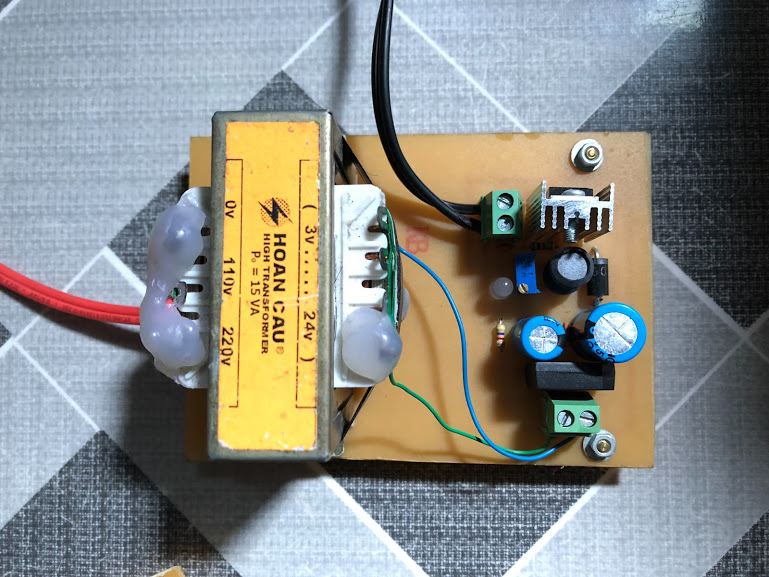
Điện áp ở dạng xung vuông với chu kỳ Tp, độ rộng Ton chính là thời gian xung ở điện áp đỉnh Vpk (Ton <= Tp). Xung vuông này sau khi cho qua mạch lọc LC sẽ bị san phẳng thành thành điện áp một chiều có giá trị Vout như hình trên. Ta có thể điều chỉnh điện áp Vout theo ý mình bằng cách điều chỉnh độ rộng xung Ton (ở đây dùng biến trở), Ton càng lớn thì Vout càng lớn và ngược lại.

c. Sơ đồ layout và mạch in



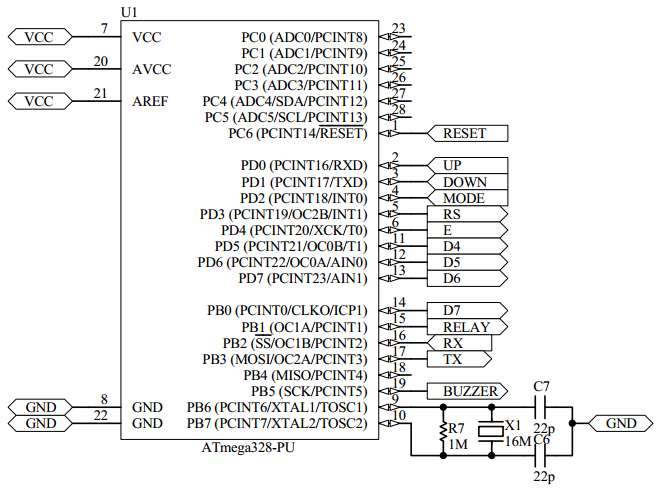
**Hình 3.4:** Sơ đồ layout của khối nguồn

**Hình 3.5:** Sơ đồ mạch in của khối nguồn

d. Hình ảnh thực tế

**Hình 3.6:** Hình ảnh thực tế của khối nguồn

### **3.2.2.** Khối vi điều khiển

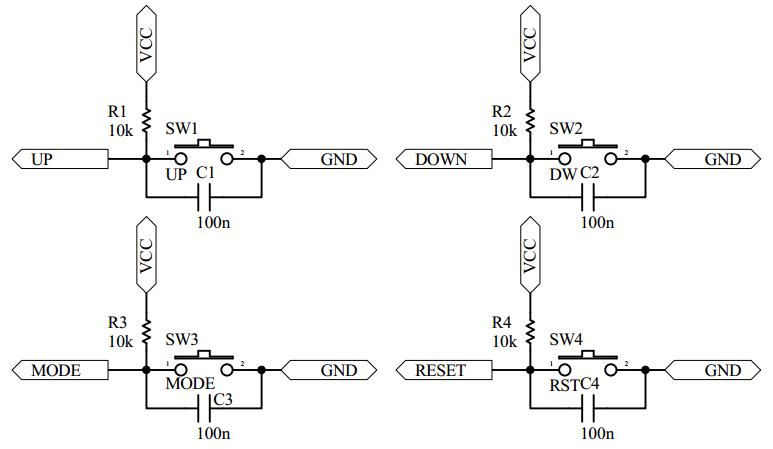
 a. Sơ đồ nguyên lý

**Hình 3.7:** Sơ đồ nguyên lý của khối vi điều khiển

Tham khảo datasheet, phụ lục trang 44.

b. Nguyên lý hoạt động: Khối vi điều khiển này sẽ nhận các dữ liệu, thông số của điện năng và gửi những dữ liệu đó cho các khối hiển thị, khối đóng ngắt tải và khối chuông.

### **3.2.3.** Khối nút nhấn

 a. Sơ đồ nguyên lý

**Hình 3.8:** Sơ đồ nguyên lý của các nút nhấn

Các thông số của linh kiện:

* Điện trở 10kΩ: Đây là điện trở kéo lên, có tác dụng hạn dòng từ VCC đến GND khi nhấn nút, tránh gây hiện tượng ngắn mạch, đảm bảo an toàn cho vi điều khiển và có tác dụng loại bỏ hiện tượng trôi nổi điện áp ở ngõ vào.
* Tụ điện 100nF: Tụ điện chống nhiễu cho nút nhấn.

b. Nguyên lý hoạt động:

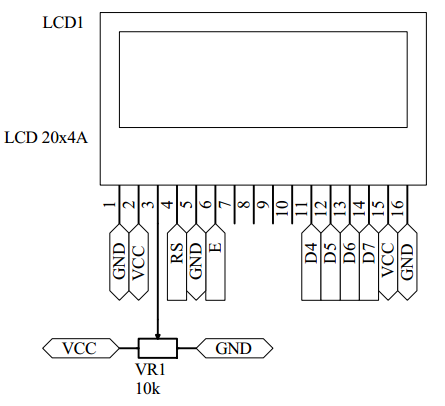
Nút nhấn RST có tác dụng reset lại dữ liệu cho toàn bộ mạch.

Nút nhấn MODE dùng để chọn chế độ hiển thị để quan sát theo yêu cầu của người dùng, dùng để chọn chế độ quan sát điện áp, dòng điện, công suất hoặc năng lượng điện tiêu thụ.

Các nút nhấn UP, DOWN dùng để thiết lập điện áp ngưỡng của điện áp, tránh hiện tượng quá áp hoặc điện áp quá thấp gây ảnh hưởng đến thiết bị điện.

**3.2.4.** Khối hiển thị

a. Sơ đồ nguyên lý



**Hình 3.9:** Sơ đồ nguyên lý của khối hiển thị LCD

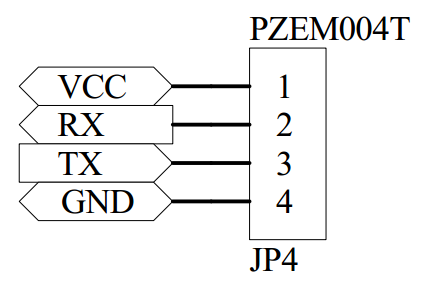
Các thông số của linh kiện:

* LCD 20x4: Dùng để hiển thị các thông số của điện năng, ở đây dùng màn hình kích thước 20x4 để hiển thị nhiều nội dung hơn.
* Biến trở 10kΩ: Dùng để điều chỉnh độ sáng của màn hình

b. Nguyên lý hoạt động

Khối hiển thị LCD sử dụng nguyên lý truyền dữ liệu 4 bit. Khi nhận dữ liệu từ khối vi điều khiển, các mọi thông số của điện năng sẽ được hiển thị trên màn hình LCD với phông chữ to sẽ giúp người dùng quan sát ở khoảng cách xa.

### **3.2.5.** Khối đo điện năng

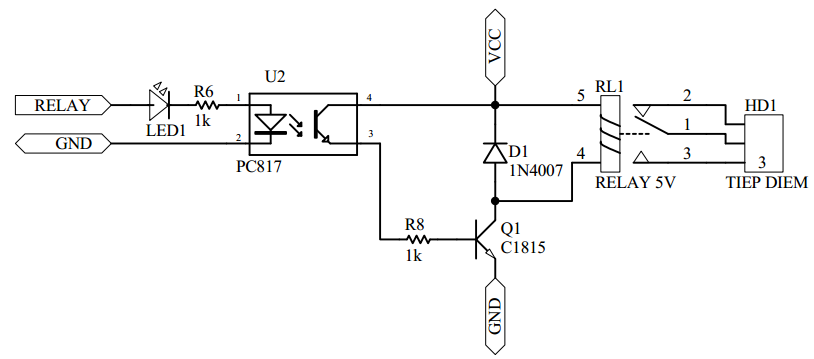
 a. Sơ đồ nguyên lý

**Hình 3.10:** Sơ đồ nguyên lý của khối đo điện năng

b. Nguyên lý hoạt động: Module đo điện năng PZEM004T sẽ được kết nối với header JP4 và giao tiếp khối vi điều khiển qua chuẩn UART.

### **3.2.6.** Khối đóng cắt tải

a. Sơ đồ nguyên lý



**Hình 3.11:** Sơ đồ nguyên lý của khối đóng ngắt tải

Các thông số của linh kiện:

* Opto PC817: Dùng để cách ly giữa vi điều khiển với relay.
* Điện trở 1kΩ (R6): Dùng để hạn dòng cho led 1 và led bên trong opto.
* Transistor C1815 Dùng để khuếch đại dòng để kích cho relay.

Dòng kích dẫn cho relay là

* Dùng dòng

Khi Transistor dẫn:

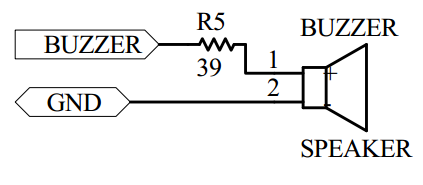
)

* Chọn

* Chọn R = 1 k dùng để hạn dòng cho chân B của transistor C1815.
* Diot 1N4007: Dùng để dập tắt điện áp tự cảm sinh ra trên cuộn dây trong relay khi đóng cắt để bảo vệ tiếp giáp CE của transistor.
* Relay 5V: Do sử dụng chung nguồn 5V với vi điều khiển nên chọn relay 5V thay vì 12V, dùng để đóng cắt điện áp khi điện áp vượt ngưỡng hoặc quá thấp.

b. Nguyên lý hoạt động: Ban đầu, nguồn điện cần đo được nối với module PZEM004T thông qua cặp tiếp điểm thường đóng của relay. Khi nhận tín hiệu từ vi điều khiển, relay hoạt động, cặp tiếp điểm thường đóng này sẽ mở ra, ngắt tải và bảo vệ thiết bị điện khi có sự cố điện áp thất thường.

### **3.2.7.** Khối chuông

 a. Sơ đồ nguyên lý

**Hình 3.12:** Sơ đồ nguyên lý của khối chuông

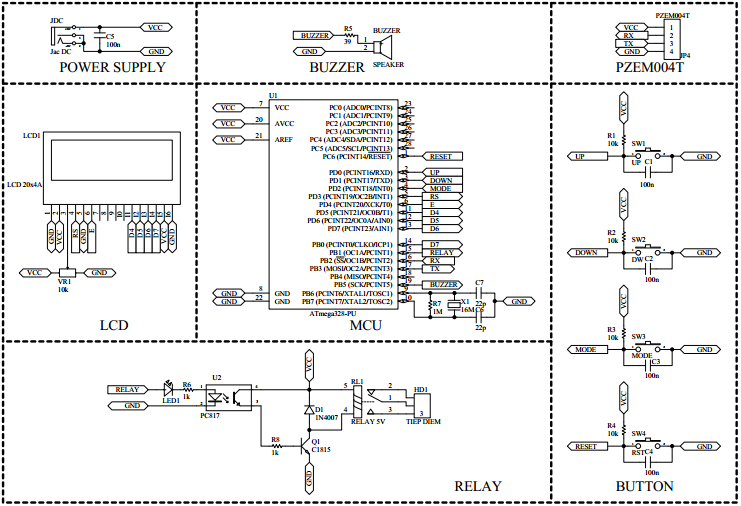
Các thông số của linh kiện:

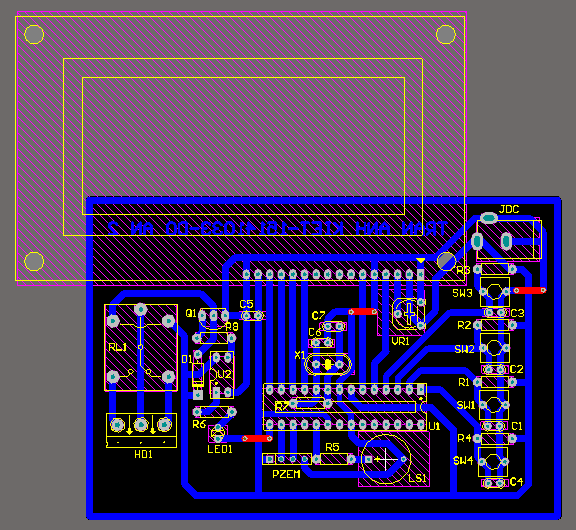
* Điện trở 39Ω: Dùng để hạn dòng cho buzzer.
* Buzzer: Vì sử dụng chung nguồn 5V với vi điều khiển nên chọn buzzer 5V, dùng để báo hiệu khi điện áp thất thường, hoạt động song song với khối đóng cắt tải.

b. Nguyên lý hoạt động: Tương tự như khối đóng cắt tải, khối chuông sẽ nhận tín hiệu từ khối vi điều khiển, chuông sẽ báo hiệu khi có sự cố điện áp thất thường giúp người dùng dễ dàng nhận biết.

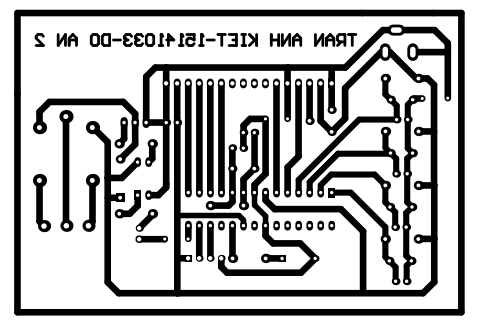
### **3.2.8.** Hệ thống hoàn chỉnh

a. Sơ đồ nguyên lý

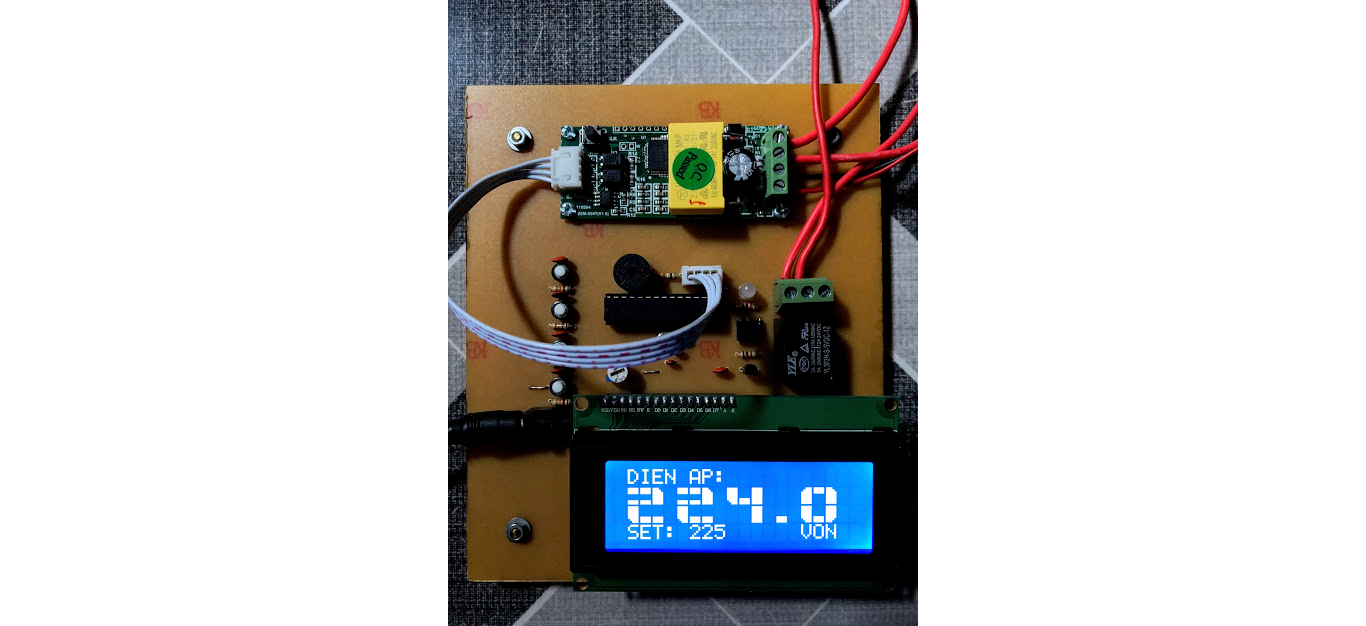
**Hình 3.13:** Sơ đồ nguyên lý của hệ thống

 b. Sơ đồ layout và mạch in

**Hình 3.14:** Sơ đồ layout của hệ thống.



**Hình 3.15:** Sơ đồ mạch in mạch của hệ thống

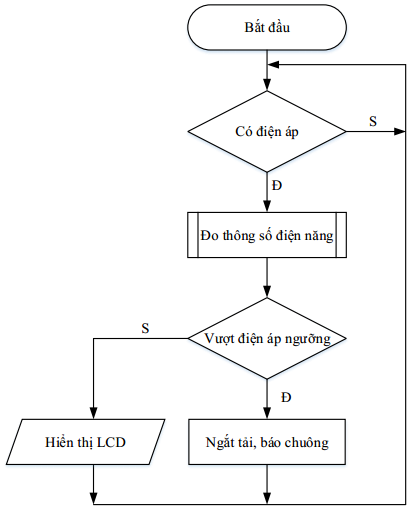
 c. Hình ảnh thực tế của hệ thống

**Hình 3.16:** Hình ảnh thực tế của hệ thống

**3.3.** Chương trình đo và giám sát điện năng

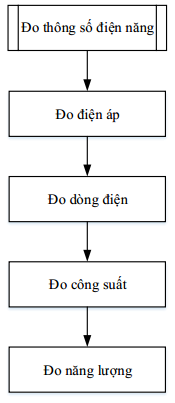
**3.3.1.** Lưu đồ đo điện năng

a. Chương trình chính



**Hình 3.17:** Chương trình chính

b. Chương trình con đo thông số điện năng



**Hình 3.18:** Chương trình con đo các thông số điện năng

d. Nguyên lý hoạt động: Ban đầu, chúng ta sẽ kiểm tra có điện áp để đo hay không. Nếu nhận được điện áp thì sẽ tiến hành đo các thông số còn lại. Khi đã đo được các thông số thì vi điều khiển sẽ xử lý và đưa ra lệnh hiển thị lên LCD. Trong quá trình đo nếu điện áp thất thường thì khối đóng ngắt tải và khối chuông sẽ hoạt động.

### **3.3.2.** Chương trình

a. Chương trình con hỗ trợ

Chương trình con hỗ trợ hiển thị LCD:

LiquidCrystal**::**LiquidCrystal**(**uint8\_t rs**,** uint8\_t enable**,**

uint8\_t d0**,** uint8\_t d1**,** uint8\_t d2**,** uint8\_t d3**)**

**{**

init**(**1**,** rs**,** 255**,** enable**,** d4**,** d5**,** d6**,** d7**,** 0**,** 0**,** 0**,** 0**);**

**}**

void LiquidCrystal**::**init**(**uint8\_t fourbitmode**,** uint8\_t rs**,** uint8\_t rw**,** uint8\_t enable**,**

uint8\_t d0**,** uint8\_t d1**,** uint8\_t d2**,** uint8\_t d3**,**

uint8\_t d4**,** uint8\_t d5**,** uint8\_t d6**,** uint8\_t d7**)**

**{**

\_rs\_pin **=** rs**;**

\_rw\_pin **=** rw**;**

\_enable\_pin **=** enable**;**

\_data\_pins**[**0**]** **=** d0**;**

\_data\_pins**[**1**]** **=** d1**;**

\_data\_pins**[**2**]** **=** d2**;**

\_data\_pins**[**3**]** **=** d3**;**

\_data\_pins**[**4**]** **=** d4**;**

\_data\_pins**[**5**]** **=** d5**;**

\_data\_pins**[**6**]** **=** d6**;**

\_data\_pins**[**7**]** **=** d7**;**

**if** **(**fourbitmode**)**

\_displayfunction **=** LCD\_4BITMODE **|** LCD\_1LINE **|** LCD\_5x8DOTS**;**

**else**

\_displayfunction **=** LCD\_8BITMODE **|** LCD\_1LINE **|** LCD\_5x8DOTS**;**

begin**(**16**,** 1**);**

**}**

void LiquidCrystal**::**begin**(**uint8\_t cols**,** uint8\_t lines**,** uint8\_t dotsize**)** **{**

**if** **(**lines **>** 1**)** **{**

\_displayfunction **|=** LCD\_2LINE**;**

**}**

\_numlines **=** lines**;**

setRowOffsets**(**0x00**,** 0x40**,** 0x00 **+** cols**,** 0x40 **+** cols**);**

// for some 1 line displays you can select a 10 pixel high font

**if** **((**dotsize **!=** LCD\_5x8DOTS**)** **&&** **(**lines **==** 1**))** **{**

\_displayfunction **|=** LCD\_5x10DOTS**;**

**}**

pinMode**(**\_rs\_pin**,** OUTPUT**);**

// we can save 1 pin by not using RW. Indicate by passing 255 instead of pin#

**if** **(**\_rw\_pin **!=** 255**)** **{**

pinMode**(**\_rw\_pin**,** OUTPUT**);**

**}**

pinMode**(**\_enable\_pin**,** OUTPUT**);**

// Do these once, instead of every time a character is drawn for speed reasons.

**for** **(**int i**=**0**;** i**<((**\_displayfunction **&** LCD\_8BITMODE**)** **?** 8 **:** 4**);** **++**i**)**

**{**

pinMode**(**\_data\_pins**[**i**],** OUTPUT**);**

**}**

// SEE PAGE 45/46 FOR INITIALIZATION SPECIFICATION!

// according to datasheet, we need at least 40ms after power rises above 2.7V

// before sending commands. Arduino can turn on way before 4.5V so we'll wait 50

delayMicroseconds**(**50000**);**

// Now we pull both RS and R/W low to begin commands

digitalWrite**(**\_rs\_pin**,** LOW**);**

digitalWrite**(**\_enable\_pin**,** LOW**);**

**if** **(**\_rw\_pin **!=** 255**)** **{**

digitalWrite**(**\_rw\_pin**,** LOW**);**

**}**

//put the LCD into 4 bit or 8 bit mode

**if** **(!** **(**\_displayfunction **&** LCD\_8BITMODE**))** **{**

// this is according to the hitachi HD44780 datasheet

// figure 24, pg 46

// we start in 8bit mode, try to set 4 bit mode

write4bits**(**0x03**);**

delayMicroseconds**(**4500**);** // wait min 4.1ms

// second try

write4bits**(**0x03**);**

delayMicroseconds**(**4500**);** // wait min 4.1ms

// third go!

write4bits**(**0x03**);**

delayMicroseconds**(**150**);**

// finally, set to 4-bit interface

write4bits**(**0x02**);**

**}** **else** **{**

// this is according to the hitachi HD44780 datasheet

// page 45 figure 23

// Send function set command sequence

command**(**LCD\_FUNCTIONSET **|** \_displayfunction**);**

delayMicroseconds**(**4500**);** // wait more than 4.1ms

// second try

command**(**LCD\_FUNCTIONSET **|** \_displayfunction**);**

delayMicroseconds**(**150**);**

// third go

command**(**LCD\_FUNCTIONSET **|** \_displayfunction**);**

**}**

// finally, set # lines, font size, etc.

command**(**LCD\_FUNCTIONSET **|** \_displayfunction**);**

// turn the display on with no cursor or blinking default

\_displaycontrol **=** LCD\_DISPLAYON **|** LCD\_CURSOROFF **|** LCD\_BLINKOFF**;**

display**();**

// clear it off

clear**();**

// Initialize to default text direction (for romance languages)

\_displaymode **=** LCD\_ENTRYLEFT **|** LCD\_ENTRYSHIFTDECREMENT**;**

// set the entry mode

command**(**LCD\_ENTRYMODESET **|** \_displaymode**);**

**}**

void LiquidCrystal**::**clear**()**

**{**

command**(**LCD\_CLEARDISPLAY**);** // clear display, set cursor position to zero

delayMicroseconds**(**2000**);** // this command takes a long time!

**}**

void LiquidCrystal**::**setCursor**(**uint8\_t col**,** uint8\_t row**)**

**{**

const size\_t max\_lines **=** **sizeof(**\_row\_offsets**)** **/** **sizeof(\***\_row\_offsets**);**

**if** **(** row **>=** max\_lines **)** **{**

row **=** max\_lines **-** 1**;** // we count rows starting w/0

**}**

**if** **(** row **>=** \_numlines **)** **{**

row **=** \_numlines **-** 1**;** // we count rows starting w/0

**}**

command**(**LCD\_SETDDRAMADDR **|** **(**col **+** \_row\_offsets**[**row**]));**

**}**

void LiquidCrystal**::**createChar**(**uint8\_t location**,** uint8\_t charmap**[])** **{**

location **&=** 0x7**;** // we only have 8 locations 0-7

command**(**LCD\_SETCGRAMADDR **|** **(**location **<<** 3**));**

**for** **(**int i**=**0**;** i**<**8**;** i**++)** **{**

write**(**charmap**[**i**]);**

**}**

**}**

inline size\_t LiquidCrystal**::**write**(**uint8\_t value**)** **{**

send**(**value**,** HIGH**);**

**return** 1**;** // assume sucess

**}**

Chương trình con hỗ trợ đo điện năng:

PZEM004T**::**PZEM004T**(**uint8\_t receivePin**,** uint8\_t transmitPin**)**

**{**

SoftwareSerial **\***port **=** **new** SoftwareSerial**(**receivePin**,** transmitPin**);**

port**->**begin**(**PZEM\_BAUD\_RATE**);**

**this->**serial **=** port**;**

**this->**\_isSoft **=** **true;**

**}**

float PZEM004T**::**voltage**(**const IPAddress **&**addr**)**

**{**

uint8\_t data**[**RESPONSE\_DATA\_SIZE**];**

send**(**addr**,** PZEM\_VOLTAGE**);**

**if(!**recieve**(**RESP\_VOLTAGE**,** data**))**

**return** PZEM\_ERROR\_VALUE**;**

**return** **(**data**[**0**]** **<<** 8**)** **+** data**[**1**]** **+** **(**data**[**2**]** **/** 10.0**);**

**}**

float PZEM004T**::**current**(**const IPAddress **&**addr**)**

**{**

uint8\_t data**[**RESPONSE\_DATA\_SIZE**];**

send**(**addr**,** PZEM\_CURRENT**);**

**if(!**recieve**(**RESP\_CURRENT**,** data**))**

**return** PZEM\_ERROR\_VALUE**;**

**return** **(**data**[**0**]** **<<** 8**)** **+** data**[**1**]** **+** **(**data**[**2**]** **/** 100.0**);**

**}**

float PZEM004T**::**power**(**const IPAddress **&**addr**)**

**{**

uint8\_t data**[**RESPONSE\_DATA\_SIZE**];**

send**(**addr**,** PZEM\_POWER**);**

**if(!**recieve**(**RESP\_POWER**,** data**))**

**return** PZEM\_ERROR\_VALUE**;**

**return** **(**data**[**0**]** **<<** 8**)** **+** data**[**1**];**

**}**

float PZEM004T**::**energy**(**const IPAddress **&**addr**)**

**{**

uint8\_t data**[**RESPONSE\_DATA\_SIZE**];**

send**(**addr**,** PZEM\_ENERGY**);**

**if(!**recieve**(**RESP\_ENERGY**,** data**))**

**return** PZEM\_ERROR\_VALUE**;**

**return** **((**uint32\_t**)**data**[**0**]** **<<** 16**)** **+** **((**uint16\_t**)**data**[**1**]** **<<** 8**)** **+** data**[**2**];**

**}**

bool PZEM004T**::**setAddress**(**const IPAddress **&**newAddr**)**

**{**

send**(**newAddr**,** PZEM\_SET\_ADDRESS**);**

**return** recieve**(**RESP\_SET\_ADDRESS**);**

**}**

Chương trình con hỗ trợ font chữ to:

byte LT**[**8**]** **=**

**{**

B00111**,**

B01111**,**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11111

**};**

byte UB**[**8**]** **=**

**{**

B11111**,**

B11111**,**

B11111**,**

B00000**,**

B00000**,**

B00000**,**

B00000**,**

B00000

**};**

byte RT**[**8**]** **=**

**{**

B11100**,**

B11110**,**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11111

**};**

byte LL**[**8**]** **=**

**{**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B01111**,**

B00111

**};**

byte LB**[**8**]** **=**

**{**

B00000**,**

B00000**,**

B00000**,**

B00000**,**

B00000**,**

B11111**,**

B11111**,**

B11111

**};**

byte LR**[**8**]** **=**

**{**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11110**,**

B11100

**};**

byte MB**[**8**]** **=**

**{**

B11111**,**

B11111**,**

B11111**,**

B00000**,**

B00000**,**

B00000**,**

B11111**,**

B11111

**};**

byte BLOCK**[**8**]** **=**

**{**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11111**,**

B11111

**};**

b. Chương trình chính

#include "LiquidCrystal.h"

#include "PZEM004T.h"

#include "TU\_TAO.h"

int count **=** 220**;**

int tt **=** 0**;**

const int up **=** 0**,** dw **=** 1**,** mode **=** 2**,** role **=** 9**,** loa **=** 13**;**

const int rs **=** 3**,** en **=** 4**,** d4 **=** 5**,** d5 **=** 6**,** d6 **=** 7**,** d7 **=** 8**;**

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

float v**,** i**,** p**,** e**;**

int v\_float**,** v\_float\_dv**,** v\_dv**,** v\_ch**,** v\_tr**;**

int i\_float**,** i\_float\_dv**,** i\_float\_ch**,** i\_dv**,** i\_ch**;**

int p\_float**,** p\_dv**,** p\_ch**,** p\_tr**,** p\_ng**;**

int e\_float**,** e\_float\_dv**,** e\_dv**,** e\_ch**,** e\_tr**;**

PZEM004T pzem**(**11**,** 10**);** // (RX,TX) connect to TX,RX of PZEM

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

void setup**()** **{**

Serial**.**begin**(**9600**);**

pzem**.**setAddress**(**ip**);**

pinMode**(**up**,** INPUT**);**

pinMode**(**dw**,** INPUT**);**

pinMode**(**mode**,** INPUT**);**

pinMode**(**role**,** OUTPUT**);**

pinMode**(**loa**,** OUTPUT**);**

digitalWrite**(**loa**,** LOW**);**

digitalWrite**(**role**,** LOW**);**

digitalWrite**(**mode**,** HIGH**);**

digitalWrite**(**dw**,** HIGH**);**

digitalWrite**(**up**,** HIGH**);**

lcd**.**begin**(**20**,** 4**);**

lcd**.**clear**();**

lcd**.**createChar**(**1**,** LT**);**

lcd**.**createChar**(**2**,** UB**);**

lcd**.**createChar**(**3**,** RT**);**

lcd**.**createChar**(**4**,** LL**);**

lcd**.**createChar**(**5**,** LB**);**

lcd**.**createChar**(**6**,** LR**);**

lcd**.**createChar**(**7**,** MB**);**

lcd**.**createChar**(**8**,** BLOCK**);**

**}**

void loop**()** **{**

**if** **(**digitalRead**(**up**)** **==** LOW**)**

**{**

delay**(**20**);**

**if** **(**digitalRead**(**up**)** **==** LOW**)**

**{**

count**++;**

**}**

**}** **while** **(**digitalRead**(**up**)** **==** LOW**);**

**if** **(**digitalRead**(**dw**)** **==** LOW**)**

**{**

delay**(**20**);**

**if** **(**digitalRead**(**dw**)** **==** LOW**)**

**{**

count**--;**

**}**

**}** **while** **(**digitalRead**(**dw**)** **==** LOW**);**

**if** **(**digitalRead**(**mode**)** **==** LOW**)**

**{**

delay**(**20**);**

**if** **(**digitalRead**(**mode**)** **==** LOW**)**

**{**

tt**++;**

**}**

**}** **while** **(**digitalRead**(**mode**)** **==** LOW**);**

**if** **(**tt **==** 8**)** tt **=** 0**;**

**switch** **(**tt**)**

**{**

**case** 0**:**

do\_dien\_ap**();**

**break;**

**case** 1**:**

lcd**.**clear**();**

tt**++;**

**break;**

**case** 2**:**

do\_dong\_dien**();**

**break;**

**case** 3**:**

lcd**.**clear**();**

tt**++;**

**break;**

**case** 4**:**

do\_cong\_suat**();**

**break;**

**case** 5**:**

lcd**.**clear**();**

tt**++;**

**break;**

**case** 6**:**

do\_dien\_nang**();**

**break;**

**case** 7**:**

lcd**.**clear**();**

tt**++;**

**break;**

**}**

**}**

void do\_dien\_ap**()** **{**

v **=** pzem**.**voltage**(**ip**);**

**if** **(**v **<** 0.0**)** v **=** 0.0**;**

**if** **(**v **>** 200 **&&** v **<** 215**)** **{**

digitalWrite**(**role**,** HIGH**);**

digitalWrite**(**loa**,** HIGH**);**

**}**

**else** **if** **(**v **>** count**)** **{**

digitalWrite**(**role**,** HIGH**);**

digitalWrite**(**loa**,** HIGH**);**

**}**

**else** **{**

digitalWrite**(**role**,** LOW**);**

digitalWrite**(**loa**,** LOW**);**

**}**

v\_float **=** v **\*** 10**;** //2

v\_float\_dv **=** v\_float **%** 10**;**

v\_dv **=** v\_float **/** 10 **%** 10**;**

v\_ch **=** v\_float **/** 100 **%** 10**;**

v\_tr **=** v\_float **/** 1000 **%** 10**;**

lcd**.**setCursor**(**1**,** 0**);**

lcd**.**print**(**"DIEN AP:"**);**

printDigits**(**v\_tr**,** 1**,** 1**)** **;**

printDigits**(**v\_ch**,** 5**,** 1**)** **;**

printDigits**(**v\_dv**,** 9**,** 1**)** **;**

lcd**.**setCursor**(**13**,** 2**);**

lcd**.**write**(**5**);**

printDigits**(**v\_float\_dv**,** 15**,** 1**)** **;**

lcd**.**setCursor**(**15**,** 3**);**

lcd**.**print**(**"VON"**);**

lcd**.**setCursor**(**1**,** 3**);**

lcd**.**print**(**"SET: "**);**

lcd**.**setCursor**(**6**,** 3**);**

lcd**.**print**(**count**);**

**}**

void do\_dong\_dien**()** **{**

i **=** pzem**.**current**(**ip**);**

**if** **(**i **<** 0.0**)** i **=** 0.0**;**

i\_float **=** i **\*** 100**;** //2202 0.05

i\_float\_dv **=** i\_float **%** 10**;**

i\_float\_ch **=** i\_float **/** 10 **%** 10**;**

i\_dv **=** i\_float **/** 100 **%** 10**;**

i\_ch **=** i\_float **/** 1000 **%** 10**;**

lcd**.**setCursor**(**1**,** 0**);**

lcd**.**print**(**"DONG DIEN:"**);**

printDigits**(**i\_ch**,** 1**,** 1**)** **;**

printDigits**(**i\_dv**,** 5**,** 1**)** **;**

lcd**.**setCursor**(**9**,** 2**);**

lcd**.**write**(**5**);**

printDigits**(**i\_float\_ch**,** 11**,** 1**)** **;**

printDigits**(**i\_float\_dv**,** 15**,** 1**)** **;**

lcd**.**setCursor**(**15**,** 3**);**

lcd**.**print**(**"AMPE"**);**

**}**

void do\_cong\_suat**()** **{**

p **=** pzem**.**power**(**ip**);**

**if** **(**p **<** 0.0**)** p **=** 0.0**;**

p\_float **=** p **;** //2202

p\_dv **=** p\_float **%** 10**;**

p\_ch **=** p\_float **/** 10 **%** 10**;**

p\_tr **=** p\_float **/** 100 **%** 10**;**

p\_ng **=** p\_float **/** 1000 **%** 10**;**

lcd**.**setCursor**(**1**,** 0**);**

lcd**.**print**(**"CONG SUAT:"**);**

printDigits**(**p\_ng**,** 1**,** 1**)** **;**

printDigits**(**p\_tr**,** 5**,** 1**)** **;**

printDigits**(**p\_ch**,** 9**,** 1**)** **;**

printDigits**(**p\_dv**,** 13**,** 1**)** **;**

lcd**.**setCursor**(**15**,** 3**);**

lcd**.**print**(**"WATT"**);**

**}**

void do\_dien\_nang**()** **{**

e **=** pzem**.**energy**(**ip**);**

**if** **(**e **<** 0.0**)** e **=** 0.0**;**

e\_float **=** e **\*** 10**;** //2202

e\_float\_dv **=** e\_float **%** 10**;**

e\_dv **=** e\_float **/** 10 **%** 10**;**

e\_ch **=** e\_float **/** 100 **%** 10**;**

e\_tr **=** e\_float **/** 1000 **%** 10**;**

lcd**.**setCursor**(**1**,** 0**);**

lcd**.**print**(**"TIEU THU:"**);**

printDigits**(**e\_tr**,** 1**,** 1**)** **;**

printDigits**(**e\_ch**,** 5**,** 1**)** **;**

printDigits**(**e\_dv**,** 9**,** 1**)** **;**

lcd**.**setCursor**(**13**,** 2**);**

lcd**.**write**(**5**);**

printDigits**(**e\_float\_dv**,** 15**,** 1**)** **;**

lcd**.**setCursor**(**15**,** 3**);**

lcd**.**print**(**"Wh"**);**

**}**

void custom0**(**int x**,** int y**)** **{**

lcd**.**setCursor**(**x**,** y**);**

lcd**.**write**(**1**);**

lcd**.**write**(**2**);**

lcd**.**write**(**3**);**

lcd**.**setCursor**(**x**,** y **+** 1**);**

lcd**.**write**(**4**);**

lcd**.**write**(**5**);**

lcd**.**write**(**6**);**

**}**

void custom1**(**int x**,** int y**)** **{**

lcd**.**setCursor**(**x**,** y**);**

lcd**.**write**(**2**);**

lcd**.**write**(**3**);**

lcd**.**print**(**" "**);**

lcd**.**setCursor**(**x**,** y **+** 1**);**

lcd**.**write**(**5**);**

lcd**.**write**(**8**);**

lcd**.**write**(**5**);**

**}**

void custom2**(**int x**,** int y**)** **{**

lcd**.**setCursor**(**x**,** y**);**

lcd**.**write**(**7**);**

lcd**.**write**(**7**);**

lcd**.**write**(**3**);**

lcd**.**setCursor**(**x**,** y **+** 1**);**

lcd**.**write**(**4**);**

lcd**.**write**(**5**);**

lcd**.**write**(**5**);**

**}**

void custom3**(**int x**,** int y**)** **{**

lcd**.**setCursor**(**x**,** y**);**

lcd**.**write**(**7**);**

lcd**.**write**(**7**);**

lcd**.**write**(**3**);**

lcd**.**setCursor**(**x**,** y **+** 1**);**

lcd**.**write**(**5**);**

lcd**.**write**(**5**);**

lcd**.**write**(**6**);**

**}**

void custom4**(**int x**,** int y**)** **{**

lcd**.**setCursor**(**x**,** y**);**

lcd**.**write**(**4**);**

lcd**.**write**(**5**);**

lcd**.**write**(**8**);**

lcd**.**setCursor**(**x**,** y **+** 1**);**

lcd**.**print**(**" "**);**

lcd**.**print**(**" "**);**

lcd**.**write**(**8**);**

**}**

void custom5**(**int x**,** int y**)** **{**

lcd**.**setCursor**(**x**,** y**);**

lcd**.**write**(**4**);**

lcd**.**write**(**7**);**

lcd**.**write**(**7**);**

lcd**.**setCursor**(**x**,** y **+** 1**);**

lcd**.**write**(**5**);**

lcd**.**write**(**5**);**

lcd**.**write**(**6**);**

**}**

void custom6**(**int x**,** int y**)** **{**

lcd**.**setCursor**(**x**,** y**);**

lcd**.**write**(**1**);**

lcd**.**write**(**7**);**

lcd**.**write**(**7**);**

lcd**.**setCursor**(**x**,** y **+** 1**);**

lcd**.**write**(**4**);**

lcd**.**write**(**5**);**

lcd**.**write**(**6**);**

**}**

void custom7**(**int x**,** int y**)** **{**

lcd**.**setCursor**(**x**,** y**);**

lcd**.**write**(**2**);**

lcd**.**write**(**2**);**

lcd**.**write**(**3**);**

lcd**.**setCursor**(**x**,** y **+** 1**);**

lcd**.**print**(**" "**);**

lcd**.**print**(**" "**);**

lcd**.**write**(**8**);**

**}**

void custom8**(**int x**,** int y**)** **{**

lcd**.**setCursor**(**x**,** y**);**

lcd**.**write**(**1**);**

lcd**.**write**(**7**);**

lcd**.**write**(**3**);**

lcd**.**setCursor**(**x**,** y **+** 1**);**

lcd**.**write**(**4**);**

lcd**.**write**(**5**);**

lcd**.**write**(**6**);**

**}**

void custom9**(**int x**,** int y**)** **{**

lcd**.**setCursor**(**x**,** y**);**

lcd**.**write**(**1**);**

lcd**.**write**(**7**);**

lcd**.**write**(**3**);**

lcd**.**setCursor**(**x**,** y **+** 1**);**

lcd**.**write**(**5**);**

lcd**.**write**(**5**);**

lcd**.**write**(**6**);**

**}**

void printDigits**(**int digits**,** int x**,** int y**)** **{**

**switch** **(**digits**)** **{**

**case** 0**:**

custom0**(**x**,** y**);**

**break;**

**case** 1**:**

custom1**(**x**,** y**);**

**break;**

**case** 2**:**

custom2**(**x**,** y**);**

**break;**

**case** 3**:**

custom3**(**x**,** y**);**

**break;**

**case** 4**:**

custom4**(**x**,** y**);**

**break;**

**case** 5**:**

custom5**(**x**,** y**);**

**break;**

**case** 6**:**

custom6**(**x**,** y**);**

**break;**

**case** 7**:**

custom7**(**x**,** y**);**

**break;**

**case** 8**:**

custom8**(**x**,** y**);**

**break;**

**case** 9**:**

custom9**(**x**,** y**);**

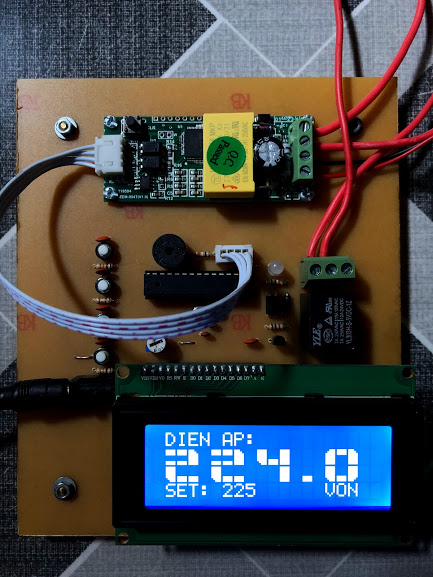
**break;**

**}**

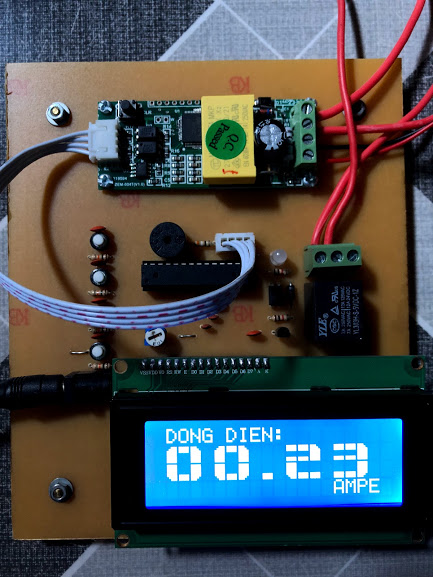
**}**

**3.3.3.** Kết quả

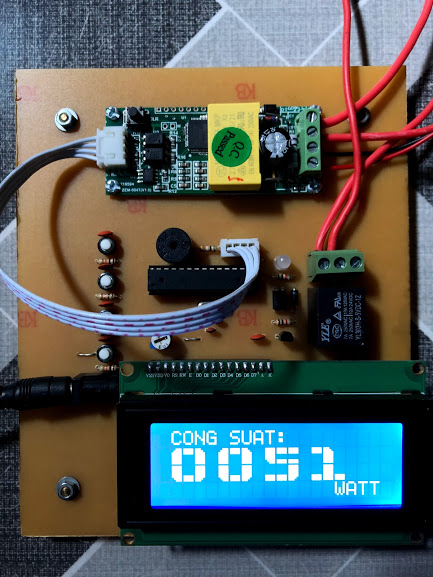
Sau khi thiết kế, tiến hành lắp ráp mạch và chạy thử thì đã đo được các thông số của điện năng như điện áp, dòng điện, công suất và năng lượng tiêu thụ cũng như việc báo động và ngắt tải khi điện áp thất thường cũng đã thực hiện tốt.



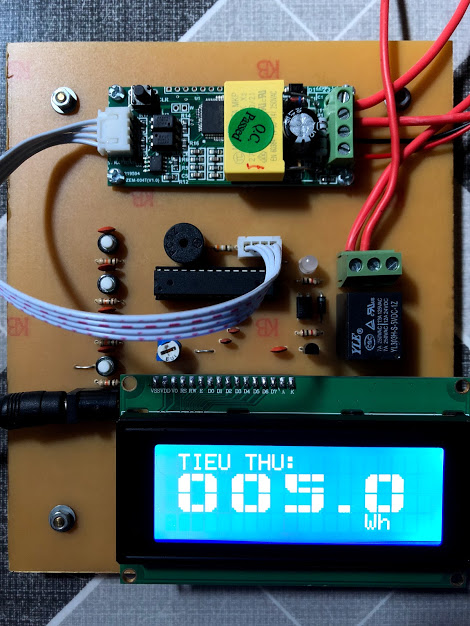
**Hình 3.19:** Kết quả đo điện áp



**Hình 3.20:** Kết quả đo dòng điện



**Hình 3.21:** Kết quả đo công suất



**Hình 3.22:** Kết quả đo năng lượng tiêu thụ

# **Chương 4**

# **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

**4.1.** Kết luận

Sau khi thực hiện đồ án môn học 2 với đề tài: **“Thiết kế hệ thống đo giám sát điện năng”** đã hoàn tất gồm các nội dung sau:

* Yêu cầu đạt được:
* Xây dựng được sơ đồ nguyên lý cho từng khối.
* Thi công được mạch thực tế.
* Mạch đo được các thông số của điện năng.
* Phân tích được các đại lượng điện.
* Những hạn chế:
* Mô hình chỉ thích hợp hoạt động trong nhà.
* Do hiển thị bằng LCD nên quan sát còn chưa rõ.

**4.2.** Hướng phát triển

Trong một nền kinh tế ngày càng phát triển của xã hội, chúng ta luôn thường xuyên tiếp xúc với điện năng, điều này hết sức nguy hiểm. Do đó em thiết kế mạch đo và giám sát này để cho chúng ta hiểu rõ hơn về các thông số điện năng cũng như giảm thiểu thiệt hại, hư hỏng thiết bị do điện áp thất thường.

Tuy bước đầu trong quá trình thiết kế mạch còn hơi thô, nhưng chúng ta cũng có thể vận dụng chúng vào việc đo và giám sát các thiết bị dân dụng dùng trong nhà như tivi, đèn, quạt… Bước cao hơn chúng ta có thể nguyên cứu phát triển mạch lên thành nhiều nút đo điện năng để phục vụ cho việc giám sát các thiết bị điện trong các nhà máy, xí nghiệp và gửi các thông số điện năng của các thiết bị đó lên Web đễ dễ dàng quản lý.

**TÀI LIỆU THAM KHẢO**

**Tiếng Việt**

1. Trần Thu Hà (2013), “Giáo trình Điện tử cơ bản”, Nhà xuất bản ĐHQG, Tp.HCM, Việt Nam.
2. Nguyễn Đình Phú, Nguyễn Trường Duy (2013), “Giáo trình Kỹ thuật số”, Nhà xuất bản ĐHQG, Tp.HCM, Việt Nam.
3. “ATMEGA328P-PU Vi điều khiển 8 bit”, <https://linhkienbandan.com/shop/atmega328-vi-dieu-khien-8-bit/>
4. “CÁC LOẠI NGUỒN XUNG THÔNG DỤNG VÀ NGUYÊN LÝ HOẠT ĐỘNG”, <https://linhkienst.com/blogs/thiet-ke-phan-cung/1000125604-cac-loai-nguon-xung-thong-dung-va-nguyen-ly-hoat-dong>
5. “MODULE ĐO THÔNG SỐ ĐIỆN XOAY CHIỀU PZEM-004”, <https://dientu360.com/module-do-thong-so-dien-xoay-chieu-pzem-004>

**Tiếng Anh**

1. “LM2596 (ACTIVE)”, <http://www.ti.com/product/LM2596>
2. “Introduction to ATmega328”, <https://www.theengineeringprojects.com/2017/08/introduction-to-atmega328.html>
3. “A short guide on how to get up and running”, <https://github.com/olehs/PZEM004T/wiki/Basic-Setup>
4. “AC digital display Multifunction Meter”, <https://forum.arduino.cc/index.php?action=dlattach;topic=480767.0;attach=220845>

**PHỤ LỤC**