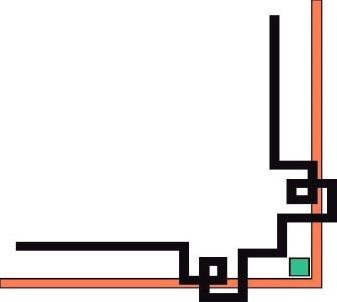
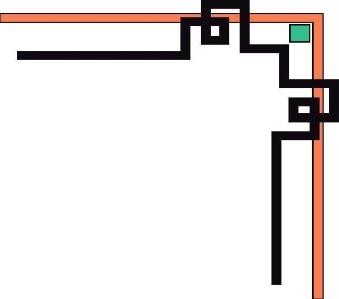
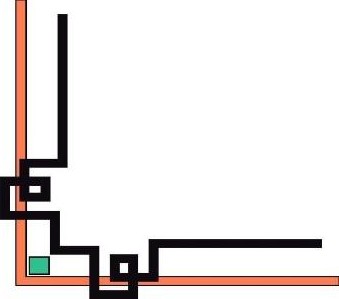
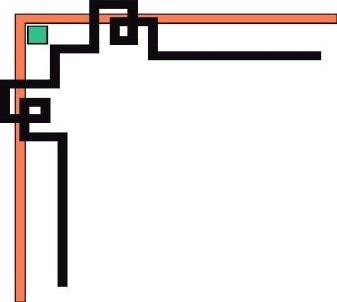
**ĐẠI HỌC QUỐC GIA HÀ NỘI**



**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

Khoa Cơ học kỹ thuật & Tự động hóa



**BÁO CÁO BÀI TẬP LỚN**

**Môn Kỹ thuật xung – số - tương tự kỹ thuật**

**đo lường & điều khiển**

**Đề tài:**

**Đọc cảm biến nhiệt độ LM35 hiển thị LCD**

**và điều khiển động cơ**

*Giáo viên hướng dẫn:* Phạm Đình Tuân

*Nhóm thực hiện:* Nhóm 10

Phạm Hoàng Hải 17020281

Ngô Văn Huy

Nguyễn Duy Hưng

Nguyễn Văn Toàn

Phùng Kim Khải

*Hà Nội, 06/2020*

**Lời mở đầu**

Đề tài này xuất phát từ ý tưởng muốn ứng dụng tự động hóa trong việc điều khiển tự động tốc độ động cơ một chiều theo nhiệt độ phòng hay nhiệt độ của thiết bị. Mạch có thể được ứng dụng trong việc tản nhiệt và điều hòa không khí. Trong quá trình làm việc, các thiết bị điện sẽ nóng dần lên, có thể ảnh hưởng đến năng suất công việc và tuổi thọ của thiết bị, nhất là đối với các thiết bị có linh kiện là vật liệu bán dẫn. Nhiệt lượng tỏa ra gây tổn hao công suất, vấn đề làm mát thiết bị là một yêu cầu tất yếu.

Có nhiều phương pháp điều khiển khác nhau như là điều khiển mạch công suất đóng cắt sử dụng Transistor, MOSFET công suất,… hay là điều khiển sử dụng khếch đại thuật toán công suất, điều khiển PID ( Bộ điều khiển tỉ lệ tích phân – vi phân),… Ngày nay, với sự phát triển của các thiết bị điện tử công suất và đặc biệt là sự phát triển nhanh chóng của vi xử lý, vi điều khiển, các bộ điều khiển số dần dần trở nên phổ biến với nhiều ưu điểm vượt trội.

Nhằm vận dụng những kiến thức đã học trên lớp về vi điều khiển và lập trình, chúng em đã nghiên cứu và thực hiện đề tài “***Đọc cảm biến nhiệt độ LM35 hiển thị LCD và điều khiển động cơ*** ” sử dụng vi điều khiển Atmega16 điều chế độ rộng xung PWM là phương pháp điều khiển tốc độ động cơ. Ngoài ra còn sử dụng truyền thông nối tiếp USART để giao tiếp gửi dữ liệu là nhiệt độ thu thập được lên máy tính qua một giao diện được xây dựng bằng GUI trên MATLAB.

**MỤC LỤC**

[**CHƯƠNG 1: CÁC KHỐI CHỨC NĂNG** 5](#_Toc43648465)

[**I.** **Mạch Nguồn** 5](#_Toc43648466)

[**1.** **Các thiết bị, linh kiện:** 5](#_Toc43648467)

[**2.** **Giới thiệu IC ổn áp dòng 78xx:** 5](#_Toc43648468)

[**3.** **Sơ đồ nguyên lý mạch:** 6](#_Toc43648469)

[**4.** **Nguyên lý hoạt động:** 7](#_Toc43648470)

[**5.** **Sơ đồ thiết kế PCB:** 8](#_Toc43648471)

[**II.** **Mạch đo lường và điều khiển** 9](#_Toc43648472)

[ **Khối điều khiển trung tâm** 9](#_Toc43648473)

[**a.** **Chức năng ADC** 11](#_Toc43648474)

[**b. Điều chế độ rộng xung PWM** 16](#_Toc43648475)

[ **Khối cảm biến LM35** 18](#_Toc43648478)

[**a. Thông số kỹ thuật** 18](#_Toc43648479)

[**b.** **Nguyên lí hoạt động** 20](#_Toc43648480)

[ **Khối Hiển Thị** 20](#_Toc43648481)

[**Giới thiệu LCD 16x2** 21](#_Toc43648482)

[**a.** **Sơ đồ chân** 21](#_Toc43648483)

[**b.** **Cách kết nối với vi điểu khiển** 22](#_Toc43648484)

[ **Khối thông báo** 23](#_Toc43648485)

[ **Khối điều khiển động cơ** 23](#_Toc43648486)

[**Giới thiệu về các linh kiện trong khối** 24](#_Toc43648487)

[**a. Opto quang** 24](#_Toc43648488)

[**Định nghĩa** 24](#_Toc43648489)

[**Nguyên lý hoạt động** 24](#_Toc43648490)

[**Ứng dụng** 25](#_Toc43648491)

[**b. MOSFET** 25](#_Toc43648492)

[**Định nghĩa:** 25](#_Toc43648493)

[**Nguyên lí hoạt động** 26](#_Toc43648494)

[**c.** **Động cơ quạt DC 12V** 28](#_Toc43648495)

[**Cấu tạo và nguyên lý làm việc của động cơ một chiều** 28](#_Toc43648496)

[ **Nguyên lý hoạt động** 29](#_Toc43648497)

[ **Khối giao tiếp với mạch tính** 30](#_Toc43648498)

[**CHƯƠNG 2: THIẾT KẾ PHẦN MỀM** 34](#_Toc43648499)

[**I.** **Thuật toán** 34](#_Toc43648500)

[**II.** **Code hoàn chỉnh** 34](#_Toc43648501)

[**CHƯƠNG 3: KẾT LUẬN, ĐÁNH GIÁ ĐỀ TÀI VÀ HƯỚNG PHÁT TRIỂN** 42](#_Toc43648502)

[ **Đánh giá** 42](#_Toc43648503)

[1. **Ưu điểm** 42](#_Toc43648504)

[**2.** **Nhược điểm** 42](#_Toc43648505)

[ **Hướng phát triển** 42](#_Toc43648503)

[**TÀI LIỆU THAM KHẢO** 43](#_Toc43648506)

**CHƯƠNG 1: CÁC KHỐI CHỨC NĂNG**

1. **Mạch Nguồn**

Nguồn điện là thành phần không thể thiếu trong các mạch điện tử và nó đóng một vai trò quan trọng ảnh hưởng tới hoạt động của mạch. Các thiết bị điện tử thông dụng thường hoạt động với các mức điện áp 3.3V, 5V, 9V, 12V, 15V, ... Trong đề tài này, vi điều khiển ATMEGA16 cần nguồn nuôi 5V và động cơ một chiều 12V.

1. **Các thiết bị, linh kiện:**

|  |  |  |
| --- | --- | --- |
| Tên thiết bị, linh kiện | Thông số | Số lượng |
| Biến áp | 220V-12V | 1 |
| Cầu diode |  | 1 |
| IC 7805 |  | 1 |
| IC 7812 |  | 1 |
| Tụ hóa | 2200 | 1 |
| Tụ gốm | 100 nF | 3 |
| Header |  | 3 |

1. **Giới thiệu IC ổn áp dòng 78xx:**

Họ IC 78xx là dòng IC ổn áp tuyến tính dùng để ổn định điện áp đầu ra với điều kiện đầu vào có thể thay đổi trong một khoảng nhất định.

Tùy loại IC 78xx mà ổn áp đầu ra là bấy nhiêu (với giá trị xx là điện áp đầu ra).

Ví dụ: IC 7805 5V, IC 7812 12V, ...

* Thông số kỹ thuật:

7805: Vin min = 7V

Vin max = 30V

I out max = 1.5A

7812: Vin min = 14V

Vin max = 40V

I out max = 1.5A

* Họ IC 78xx gồm có 3 chân:

A screenshot of a cell phone

Description automatically generated

Hình 1. Cấu tạo IC 7805.

1. **Sơ đồ nguyên lý mạch:**

A close up of a map

Description automatically generated

Hình 2. Sơ đồ nguyên lý của mạch.

Trong đó:

* Cầu diode có nhiệm vụ biến đổi điện áp xoay chiều thành điện áp một chiều.
* C1, C2 lọc nguồn đầu vào.
* C3, C4 lọc nguồn đầu ra.

1. **Nguyên lý hoạt động:**

Khối mạch ổn áp

Khối mạch chỉnh lưu

Đầu ra DC

Nguồn AC

A close up of a map

Description automatically generated

Máy biến áp

* Dòng điện đi từ nguồn 220V qua máy biến áp, máy biến áp sẽ chuyển đổi điện áp từ 220VAC sang 12VAC.
* Khối mạch chỉnh lưu: Sử dụng diode cầu 1A để chỉnh lưu điện áp xoay chiều, chúng kết hợp với tụ chỉnh lưu để tạo ra điện áp DC có giá trị 12
* Khối mạch ổn áp: Có nhiệm vụ tạo điện áp ổn định 12V và 5V ở đầu ra. Sử dụng IC 7812 cho điện áp đầu ra 12V, IC 7805 cho ra điện áp 5V. IC cho dòng ra định danh là 1.5A tuy nhiên trên thực tế thì dòng ra khoảng 500mA.

1. **Sơ đồ thiết kế PCB:**

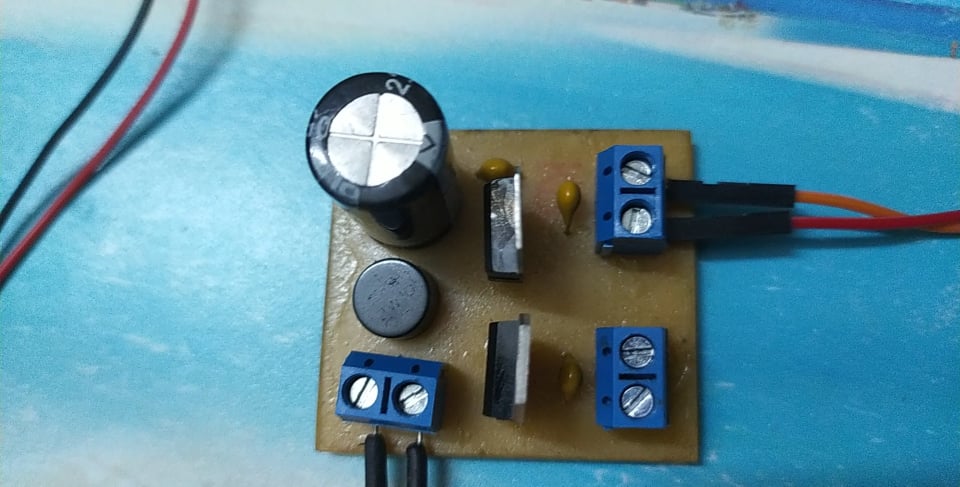
* Sơ đồ mạch in:

A screen shot of a city

Description automatically generated

Hình 4. Sơ đồ mạch in thiết kế bằng Altium.

* Hình ảnh mạch thực tế:



Hình 5. Hình ảnh thực tế mạch nguồn

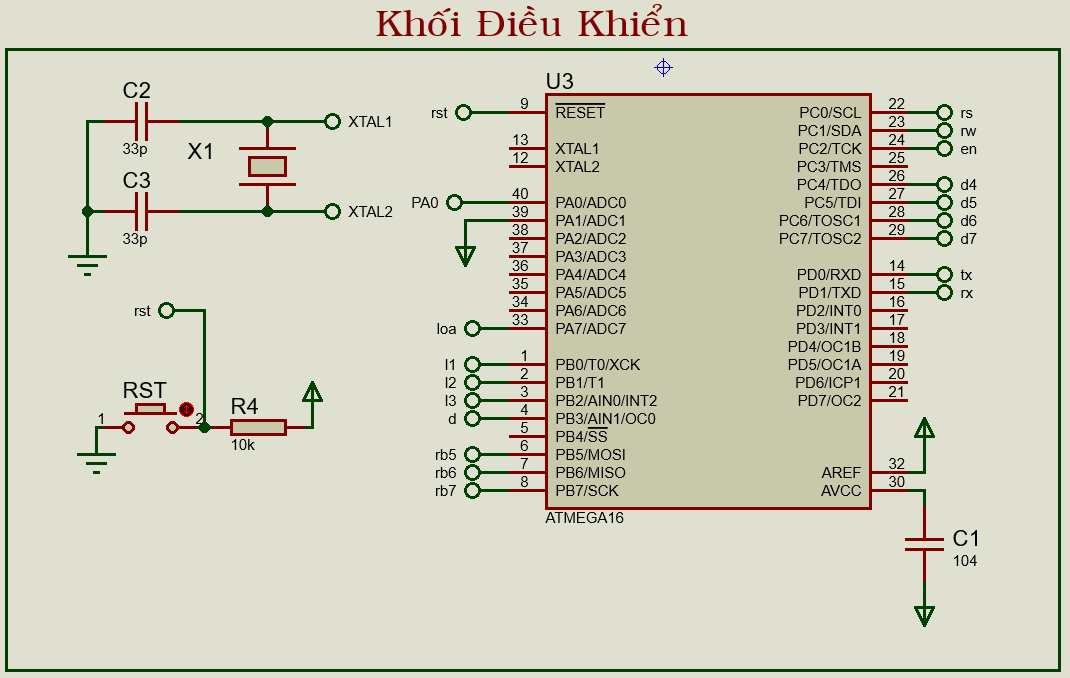
****

Hình 6. Ảnh mặt đi dây của mạch nguồn

1. **Mạch đo lường và điều khiển**

Mạch điều khiển được chia thành 6 khối.

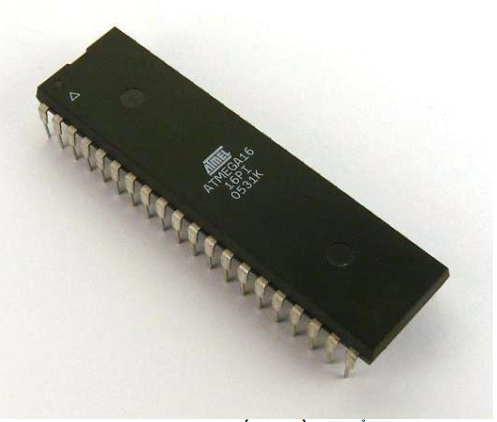
* **Khối điều khiển trung tâm**



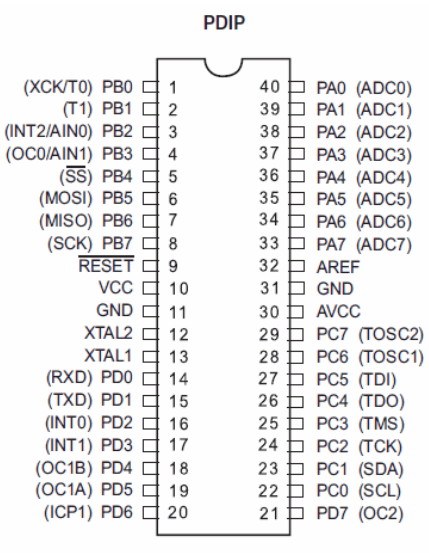
Hình 7. Khối điều khiển trung tâm

**Giới thiệu về vi điều khiển ATMEGA16**

ATmega16 là vi điều khiển 8 bit dựa trên kiến trúc RISC. Với khả năng thực hiện mỗi lệnh trong vòng một chu kỳ xung clock, ATmega16 có thể đạt được tốc độ 1MIPS trên mỗi MHz. Ngoài ra ATmega16 có các đặc điểm sau: 16KB bộ nhớ Flash với khả năng đọc trong khi ghi, 512 byte bộ nhớ EEPROM, 1KB bộ nhớ SRAM, 32 thanh ghi chức năng chung, 32 đường vào ra chung, 3 bộ định thời/bộ đếm, ngắt nội và ngắt ngoại, USART, giao tiếp nối tiếp 2 dây, 8 kênh ADC 10 bit, ... ATmega 16 hỗ trợ đầy đủ các chương trình và công cụ phát triển hệ thống như: trình dịch C, macro assemblers, chương trình mô phỏng/sửa lỗi, …



Hình 8. Hình ảnh Atmega16 trong thực tế

**

Hình 9. Sơ đồ chân của Atmega16

#### **Ý nghĩa các chân**

+ Chân VCC: Chân số 10 là VCC cấp điện áp nguồn cho Vi điều khiển. Nguồn điện cấp trong khoảng +5V ± 0,5.

+ Chân GND: Chân số11 và chân số 31 nối GND(hay nối Mass).

+ Port A (PA): Port A gồm 8 chân (từ chân 33 đến 40) có chức năng: đầu vào cho chuyển đổi ADC

+ Port B (PB): Port PB gồm 8 chân (từ chân 1 đến chân 8), ngoài có chức năng làm các đường xuất/nhập thì còn có nhiều chức năng phụ khác.

+ Port C (PC): Port C gồm 8 chân (từ chân 22 đến chân 29) : Nếu giao tiếp JTAG được kích hoạt điện trở trên các PC5(TDI), PC3 (TMS) ,PC2 (TCK) sẽ được kích hoạt ngay cả khi khởi động lại (reset)

+ Port D (PD): Port D gồm 8 chân (từ chân 14 đến 21):chưc năng xuất nhập + Chân RESET(RST): Ngõ vào RST ở chân 9 là ngõ vào Reset dùng để thiết lập trạng thái ban đầu cho vi điều khiển. Hệ thống sẽ được thiết lập lại các giá trị ban đầu nếu ngõ này ở mức 1 tối thiểu 2 chu kì máy.

+ Chân XTAL1 và XTAL2: Hai chân này có vị trí chân là 12 và 13 được sử dụng để nhận nguồn xung clock từ bên ngoài để hoạt động, thường được ghép nối với thạch anh và các tụ để tạo nguồn xung clock ổn định.

+ Chân AVCC: Nguồn cấp cho cổng A và bộ chuyển đổi ADC , chân này nên được nối với nguồn cấp VCC bên ngoài , ngay cả khi bộ chuyển đổi ADC không được sử dụng. Nếu bộ chuyển đổi ADC không được sử dụng , chân AVCC nên được nối với nguồn qua bộ lọc.

+ Chân AREF: AREF là chân chuẩn analog cho bộ chuyển đổi ADC.

#### **Chức năng ADC trong Atmega16**

Vi điều khiển Atmega16 được trang bị một bộ chuyển đổi ADC linh hoạt và mạnh mẽ với các tính năng sau:

* 8 kênh đầu vào với độ phân giải 10 bit
* Độ chính xác tuyệt đối LSB
* Lựa chọn sắp xếp dữ liệu căn phải hoặc căn trái
* Dải điện áp chuyển đổi từ 0 đến Vcc

Kết quả chuyển đổi 10 bit được lưu trong thanh ghi dữ liệu ADCH/L. Hai thnh ghi này tạo thành thanh ghi dữ liệu có độ rông 16 bit được sử dụng để lưu kết quả 10 bit. Kết quả có thể được căn trái hoặc căn phải phụ thuộc vào giá trị bit ADLAR trong thanh ghi ADMUX. Khi quá trình chuyển đổi ADC hoàn tất, bit cờ ngắt ADIF có giá trị là 1

Quá trình chuyển đổi tương tự sang số bao gồm các bước sau:

1. Khởi tạo bộ chuyển đổi ADC

Trong đề tài này, giá trị chia tần là 8 ( Vi điều khiển hoạt động với xung đồng hồ là 8MHz)

|  |
| --- |
| void ADC\_Init()  {  ADMUX = 0x40; // Can le phai, dien ap tham chieu = Avcc,  // chon kenh dau vao chuyen doi ADC0  ADCSRA = 0x83; // cho phép ADC hoạt động, bộ chia tần bằng 8  } |

1. Đọc điện áp tương tự từ bộ chuyển đổi ADC

Vòng lặp while dùng để kiểm tra trạng thái bit ADIF, bit này có giá trị là 1 khi quá trình chuyển đổi hoàn tất. Sau khi chuyển đổi, bit ADIF cần được xóa bằng cách viết một giá trị logic 1 tới nó.

Kết quả được trả về ở dạng 10 bit nhị phân, căn lề trái. Bộ chia tần ADC được cài đặt bằng 8 để có kết quả chính xác. Sauk hi chuyển đổi ADC hoàn tất, kết quả trong thanh ghi 8 bit ( ADCL và ADCH) được nối vào một biến nguyên không dấu 16 bit và được trả lại.

|  |
| --- |
| int ADC\_Read()  {  ADMUX = 0x40; // Can le phai, dien ap tham chieu = Avcc,  //chon kenh dau vao chuyen doi ADC0  ADCSRA |= 0x43; // ADCSC = 1, bat dau chuyen doi, chia tần 8  while(!(ADCSRA & 0x10)); // Kiem tra chuyen doi ket thuc chua  ADCSRA |= 0x10; // Xoa co ngat bang cach ghi 1 toi bit ADIF    return ADCW;  } |

1. **PWM trong Atmega16**Trong vi điều khiển AVR Atemega16 thì chúng ta có 3 Timer T0, T1 và T2. Timer0 và Timer2 là các bộ định thời 8 bit, trong đó Timer1 là bộ định thời 16 bit. Mỗi bộ định thời được trang bị một bộ chia tần riêng. Bộ chia tần được sử dụng để chia nguồn xung đồng hồ chính của vi điều khiển thành nguồn xung đồng hồ cho các bộ định thời sử dụng. Trong các Timer chúng ta có thể set chế độ hoạt động cho chúng là:  
   + Chế độ định thời  
   + Chế độ đếm  
   + PWM

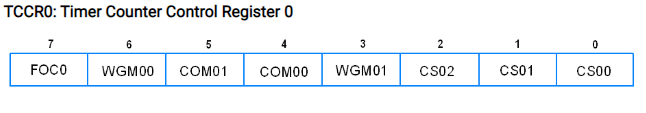
Có 2 chế độ của điều chế độ rộng xung PWM đó là:

+ Fast PWM ( PWM tần số cao)

+ Phase correct PWM ( PWM với pha chính xác)

Trong phạm vi của đề tài, chúng tôi sử dụng Timer0 được set ở chế độ PWM để điều khiển tốc độ quay của động cơ DC và sử dụng Phase correct PWM.

Sử dụng chân PB3 (OC0) – pin3 của PORT B để đưa tín hiệu điều khiển ra ngoài.



TCCRO là thanh ghi điều khiển hoạt động của Timer0:

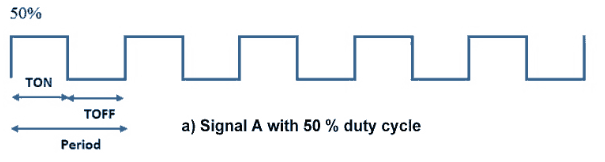
|  |
| --- |
| TCCR0 |= (1<<WGM00)|(1<<COM01)|(1<<COM00)|(1<<CS00)|(1<<CS02);  //su dung phase correct PWM, inverted PWM, bo chia 1024 |

TCNT0 và OCR0 là các thanh ghi 8 bit. Ở chế độ Phase correct PWM, thanh ghi TCNT0 sẽ đếm từ 0 đến 255 rồi từ 255 về 0. Quá trình đếm lặp đi lặp lại như vậy tạo thành các xung tam giác cân. Thanh ghi OCR0 dùng để lưu giá trị Duty cycle.



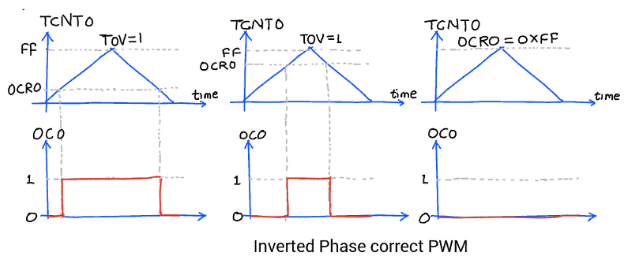
Giả sử, một xung với chu kỳ là 10ms trong đó thời gian ở mức cao(high) là 5ms thì Duty cycle sẽ được tính:

D = 5ms / 10ms = 50%



Với D = 50% ta set giá trị thanh ghi OCR 0 bằng 127. 127/255 ~ 50%

Khi mà thanh ghi TCNT0 đếm từ 0 đến 255 đếm đến đúng giá trị OCR0 thì chân OC0 đang xuất giá trị 0 sẽ lên 1 và khi TCNT0 đếm từ 255 về 0 thì đến giá trị OCR0 sẽ lại từ 1 về 0 đến hết chu kỳ. Quá trình cứ lặp đi lặp lại tại mỗi chu kỳ sẽ cho ta một dải xung vuông điều khiển như ý muốn.



Dựa vào việc thay đổi giá trị thanh ghi OCR0 chúng tôi có thể điều chỉnh tốc độ của động cơ theo 4 mức độ sau

* Khi nhiệt độ < 25°C : OCR0 = 255. Động cơ không hoạt động
* Khi 25°C <= nhiệt độ <30°C : OCR0 = 178 . Động cơ chạy mức bé nhất
* Khi 30°C <= nhiệt độ <35°C : OCR0 = 76. Động cơ chạy mức trung bình
* Khi nhiệt độ >=35: OCR0 = 0. Động cơ chạy mạnh nhất.

1. **Truyền thông nối tiếp USART**

Truyền thông nối tiếp USART cung cấp đường truyền 2 chiều giữa thiết bị truyền và nhận. Atmega16 được trang bị phần cứng với bộ truyền và nhận độc lâp với nhau. Chân truyền dữ liệu TxD (Transmitted Data – chân PD1) và chân nhận dữ liệu RxD (Received Data – chân PD0).

|  |
| --- |
| Tốc độ truyền = |

USART thường được sử dụng trong truyền thông nối tiếp không đồng bộ. Để duy trì sự không đồng bộ giữa bên truyền và nhận, các bit START và STOP được sử dụng ở đầu và cuối mỗi byte dữ liệu trong chuỗi truyền đi. Chúng ta có thể lựa chọn các tốc độ truyền dữ liệu (bit/s) khác nhau. USART cũng có thể thiết lập độ rộng dữ liệu từ 5 – 9 bit với 1 hoặc 2-bit STOP. ATmega16 cũng được trang bị phần cứng để tạo/ kiểm tra tính chẵn lẻ. Cơ chế kiểm tra bit chẵn lẻ cho phép phát hiện lỗi bit trong quá trình truyền một byte dữ liệu.

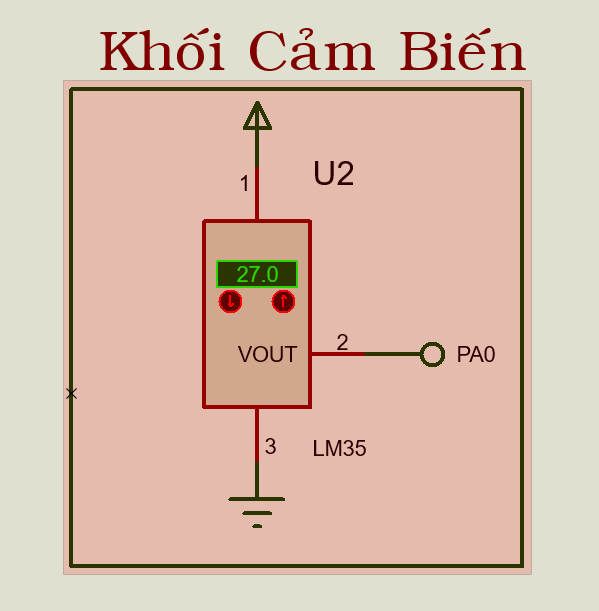
Nhóm em set cho Vi điều khiển hoạt động ở tốc độ truyền là 9600(bit/s). Chế độ hoạt động không đồng bộ, không kiểm tra chẵn lẻ, 1 bit STOP và 8 bit dữ liệu.

|  |
| --- |
| void UART\_Init() //ham khoi tao UART  {  UBRRH=0;  UBRRL=51; //Cai dat toc do Baud la 9600 voi tan so 8Mhz  UCSRA=0x00; //Khoi tao su dung UART  UCSRC=(1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0); // Set thanh ghi UCSRC, khong chan le, 8 bit du lieu, 1 bit stop  UCSRB|=(1<<RXEN)|(1<<TXEN)|(1<<RXCIE); //Khai bao truyen du lieu, nhan du lieu va ngat khi nhan |



Chuỗi dữ liệu là giá trị nhiệt độ được gửi đi

### **Khối cảm biến LM35**

****

Hình 15. Khối Cảm biến

#### **a. Thông số kỹ thuật**

+ Điện áp đầu vào từ 4V đến 30V

+ Điện áp ra: -1V đến 6V

+ Công suất tiêu thụ là 60uA

+ Độ phân giải điện áp đầu ra là 10mV/oC

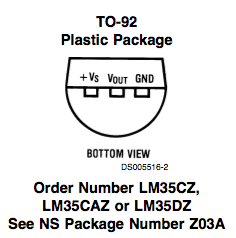
+ Độ chính xác cao ở 25 C là 0.5 C

+ Trở kháng đầu ra thấp 0.1 cho 1mA tải

+ Độ chính xác thực tế: 1/4°C ở nhiệt độ phòng và 3/4°C ngoài khoảng -55°C tới 150°C



*Hình 16. Ảnh thực tế của LM35*



*Hình 17 . Sơ đồ chân của LM35*

* Chân 1: Nối với nguồn 5V.
* Chân 2: Điện áp một chiều DC có độ phân giải 10mV/ ºC.
* Chân 3: Chân nối mass.

Để đo lường nhiệt độ thì có thể dùng nhiều loại cảm biến nhiệt khác, mỗi loại có một ưu điểm riêng phù hợp với từng nhu cầu riêng. Ở đây đề tài là đo nhiệt độ môi trường bình thường nên sử dụng LM35 là tối ưu nhất vì: đây là loại cảm biến có độ chính xác cao, tầm hoạt động tuyến tính từ 0-128 ºC, tiêu tán công suất thấp.

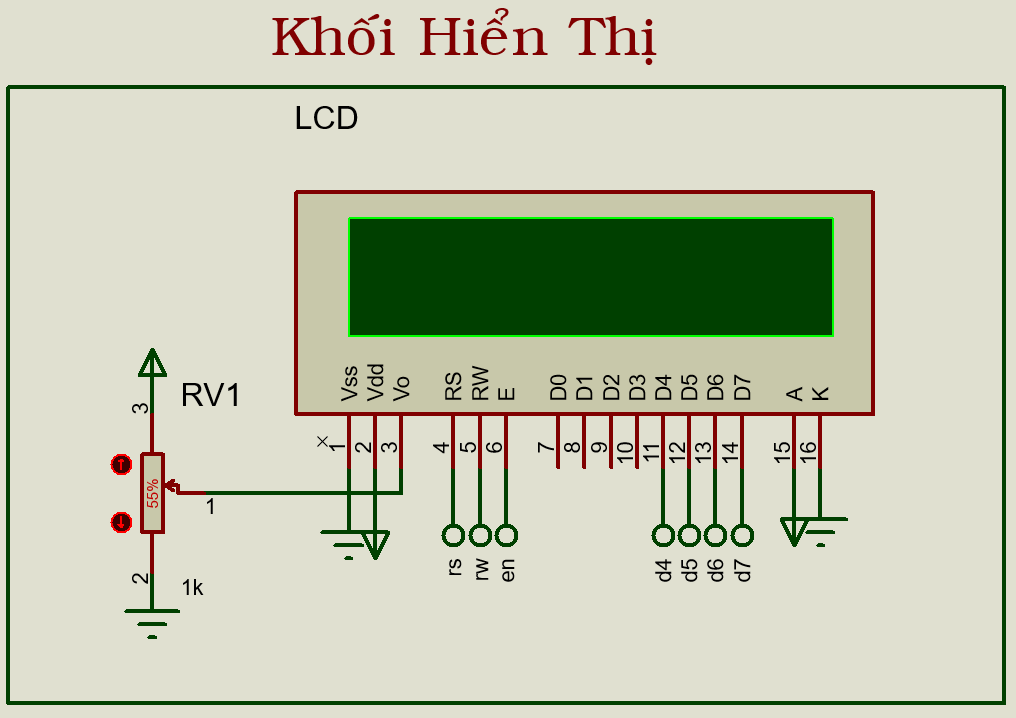
#### **Nguyên lí hoạt động**

Bộ chuyển đổi AD của Atmega16 có độ phân phải 10 bit (- 1 = 1023 giá trị). Điện áp tham chiếu là Vcc = 5V. Mỗi giá trị đầu ra của bộ ADC ứng với 5/1023 4.88mV.

Vậy độ phân giải của ADC tương ứng với 4.88mV. Do độ phân giải của LM35 là 10mV/ ºC nên ta có hệ số chuyển đổi 2.046 là lượng thay đổi trên chân ADC ứng vớ thay đổi 1 ºC trên LM35.

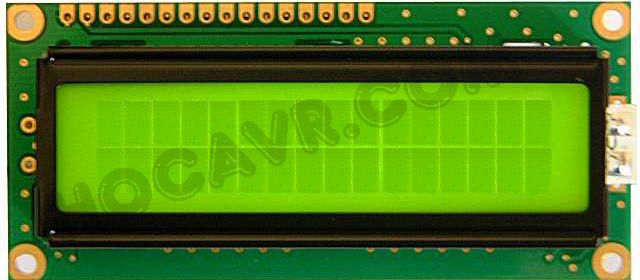
Suy ra nhiệt độ đo được: Nhiệt độ = ADC\_Read() / 2,046 (độ C)

### **Khối Hiển Thị**

****

Hình 18. Khối Hiển Thị

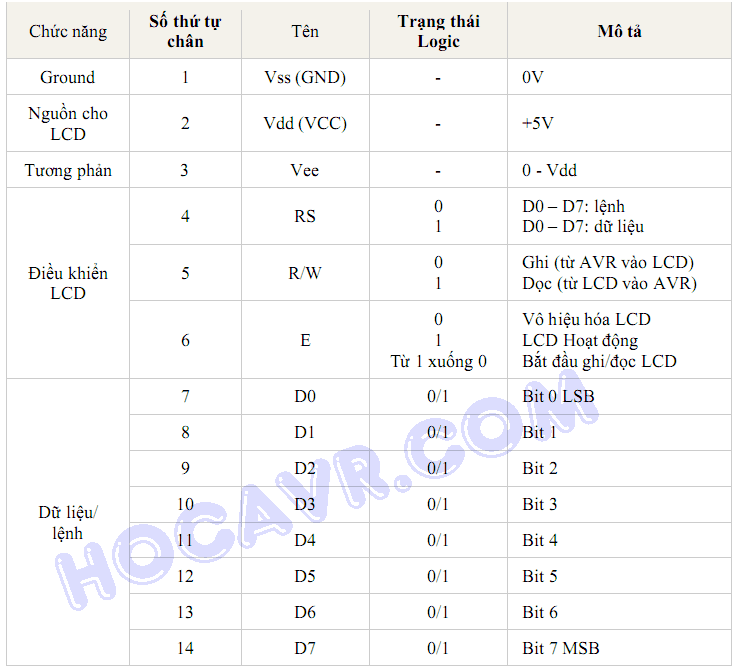
#### **Giới thiệu LCD 16x2**



Hình 19. Hình ảnh thực tế của LCD 16x2

#### **Sơ đồ chân**

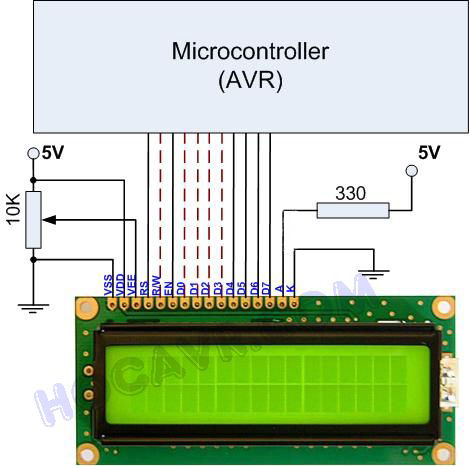
Các Text LCD theo chuẩn HD44780U thường có 16 chân trong đó 14 chân kết nối với bộ điều khiển và 2 chân nguồn cho “đèn LED nền”. Thứ tự các chân thường được sắp xếp như sau:



Bảng 1. Sơ đồ chân.

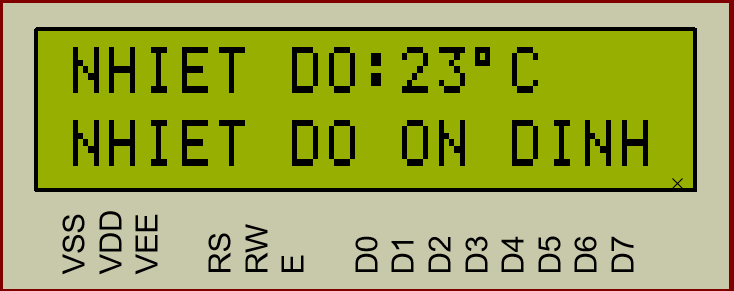
Trong một số LCD 2 chân LED nền được đánh số 15 và 16 nhưng trong một số trường hợp 2 chân này được ghi là A (Anode) và K (Cathode).Hình 11 mô tả cách kết nối LCD và nguồn mạch điều khiển.

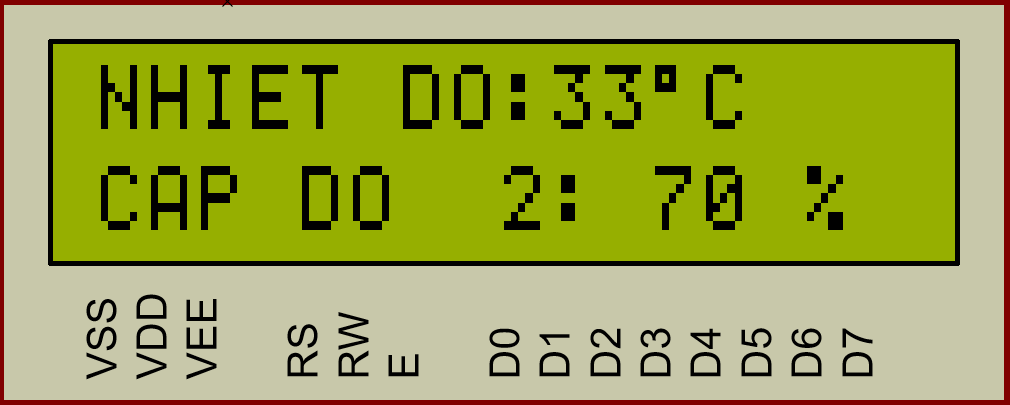
#### **Cách kết nối với vi điểu khiển**



Hình 20. Kết nối với LCD 16x2

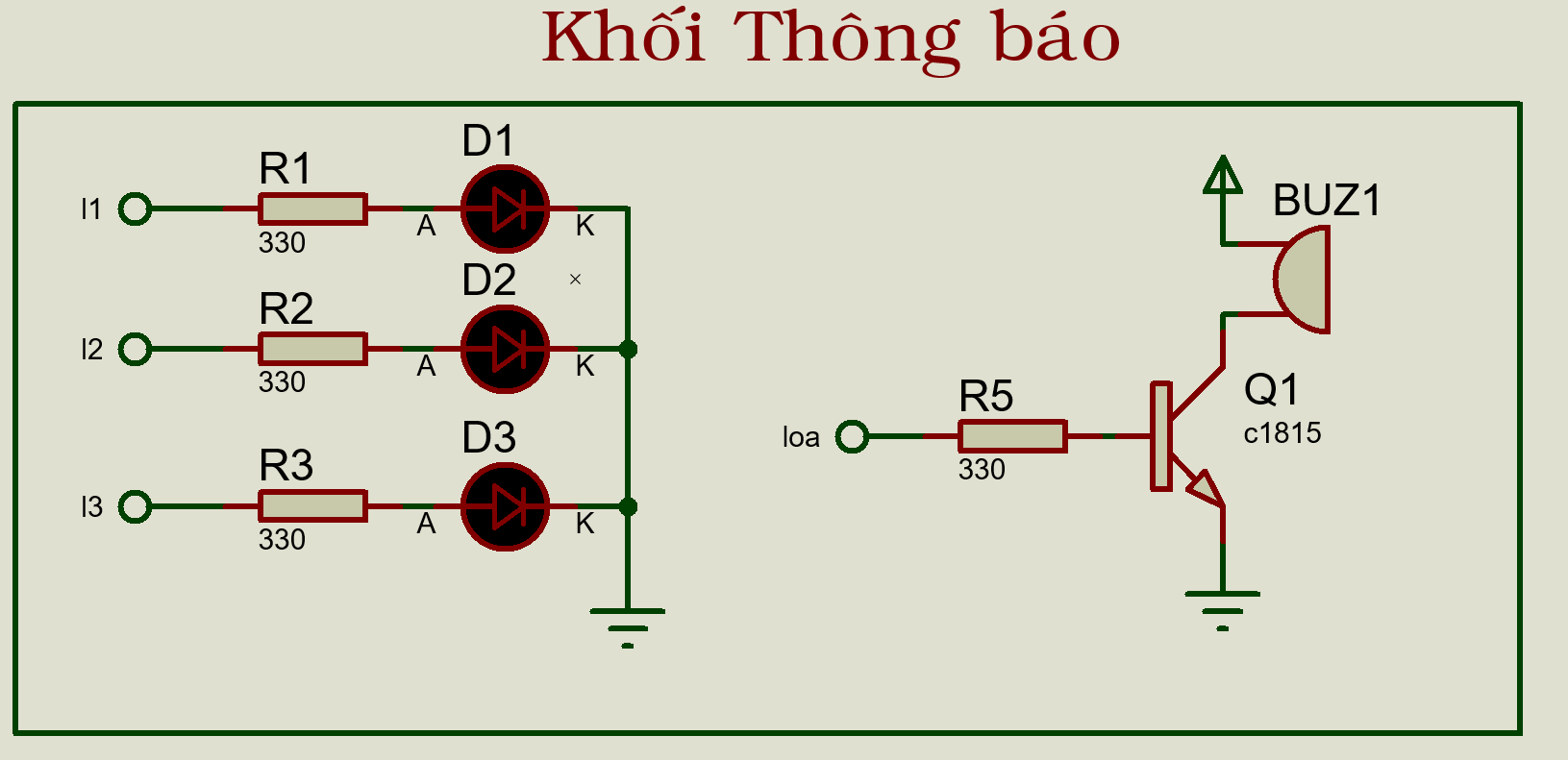
Chân 1 và chân 2 là các chân nguồn, được nối với GND và nguồn 5V. Chân 3 là chân chỉnh độ tương phản (contrast), chân này cần được nối với 1 biến trở chia áp. Trong khi hoạt động, chỉnh để thay đổi giá trị biến trở để đạt được độ tương phản cần thiết, sau đó giữ mức biến trở này. Các chân điều khiển RS, R/W, EN và các đường dữ liệu được nối trực tiếp với vi điều khiển. Ở đây chúng tôi set chế độ hoạt động là truyền 4 bit nên chỉ sử dụng 4 chân D4 đến D7 để truyền dữ liệu.





LCD hiển thi ra giá trị nhiệt độ và tốc độ quay của động cơ

### **Khối thông báo**

****

Hình 21. Khối Thông báo

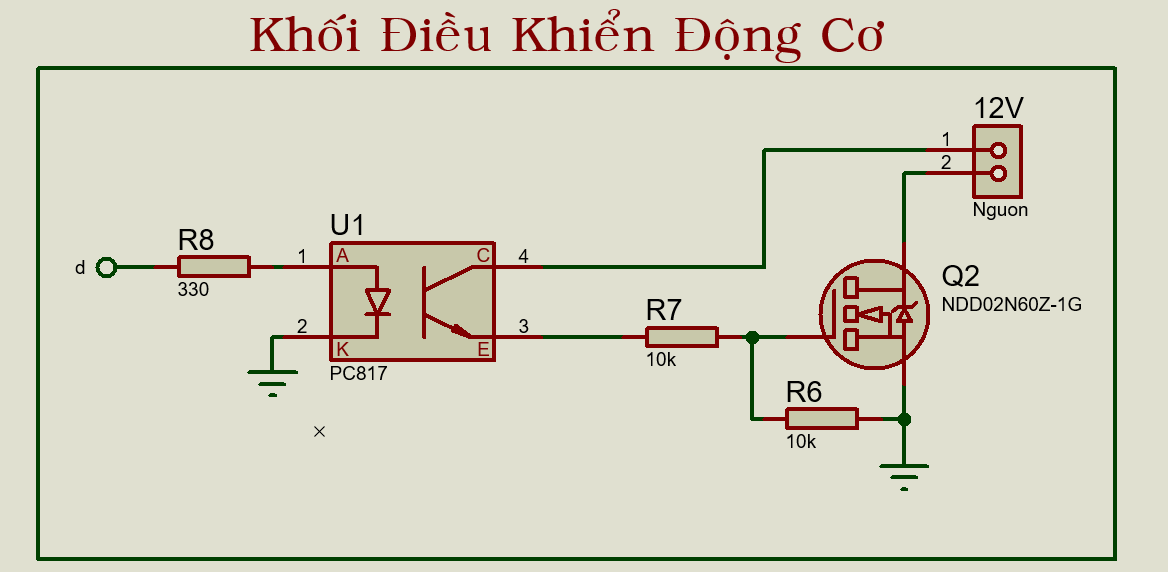
Khối thông báo bao gồm 3 đèn led biểu thị nhiệt độ ở các mức LED D1(PB0) mức 1; D2(PB1) mức 2; D3( PB2) mức 3 và có thêm còi báo buzzer(PA7) kêu khi nhiệt độ ở mức 3.

Các LED được mắc chung cực âm Katot nên tại một thời điểm cực dương của LED nào dương nhất LED sẽ sáng. Tín hiệu sẽ từ chân của VDK qua một điện trở 330Ω nhằm hạn dòng cho LED tới cực dương Anot.

Còi Buzzer được nối cực dương với nguồn 5V và cực âm được nối với chân C của Transistor C1815 nhằm kích một dòn điện lớn hơn để còi có thể hoạt động bình thường. Chân B của Transistor được nối với chân của VDK thông qua một điện trở 330Ω.

|  |
| --- |
| if(celsius >= 30 && celsius < 35)  {  sbi(PORTB,1); // Bat den 1  cbi(PORTB,0); // Tat den 0  cbi(PORTB,2); // Tat den 2  cbi(PORTA,7); // Buzzer tat  } |

### **Khối điều khiển động cơ**



Hình 22. Khối điều khiển động cơ

#### **Giới thiệu về các linh kiện trong khối**

#### **a. Opto quang (PC817)**

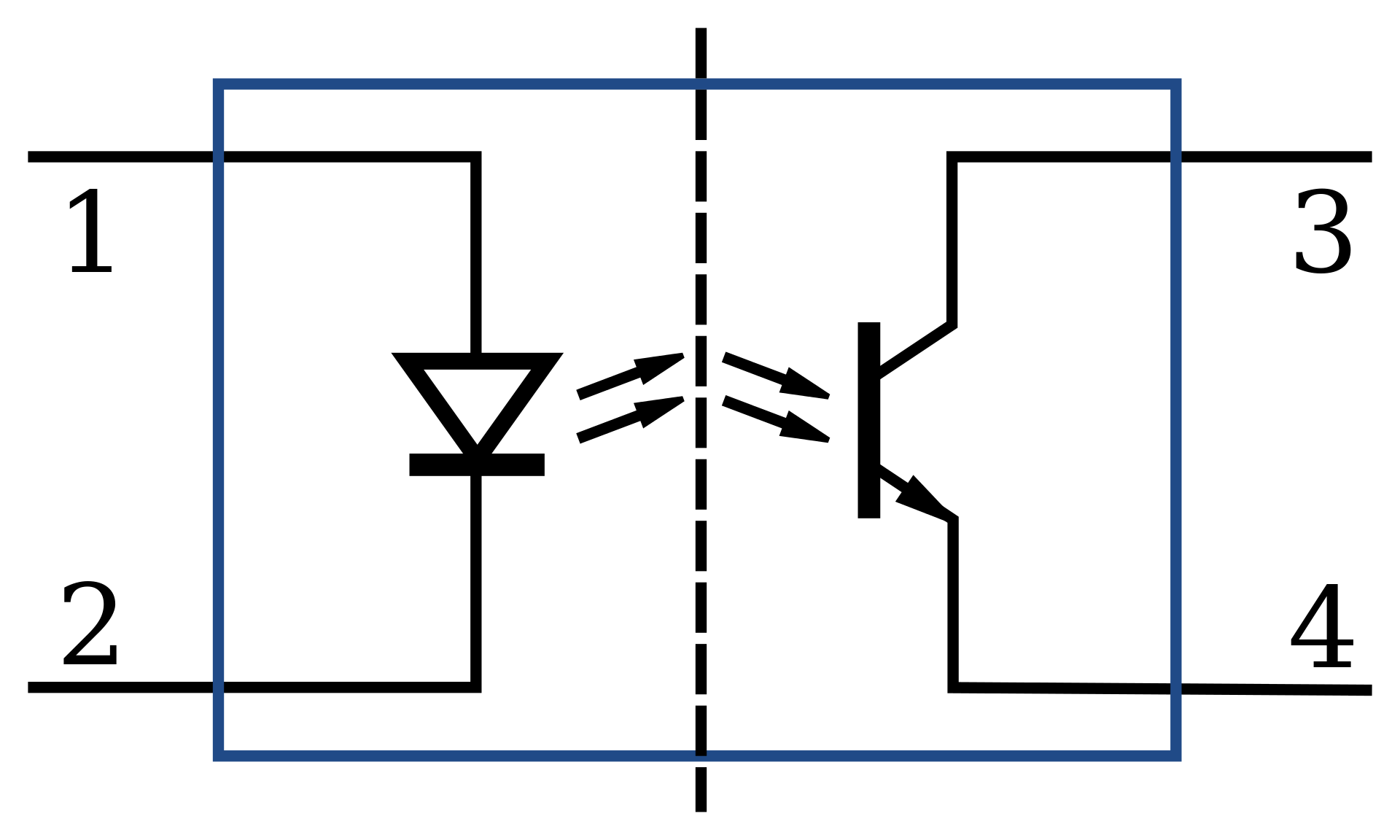
****

Hình 23. Hình ảnh thực tế của opto quang

|  |  |
| --- | --- |
| Input  (Reverse Voltage) = 6V  (Forward Voltage) = 60mA  (Forward Voltage) = 1.25 -1.6V | Output  max = 70V  max = 7V  max = 50mA |

##### **Định nghĩa**

Opto là một linh kiện quang điện tử chuyên dùng để truyền tín hiệu điều khiển giữa hai mạch điện có sự chênh lệch cao về điện áp thông qua ánh sáng mà không cần liên hệ với nhau bằng tín hiệu điện. Một opto cơ bản có cấu trúc bao gồm một đèn LED và một photo transistor bên trong nó.  Chính vì sử dụng nguyên tắc dùng ánh sáng để truyền tín hiệu lên nó còn được gọi là opto quang.



Hình 24. Kí hiệu điện của opto quang

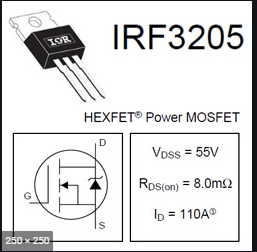
trong opto có hai thành phần bao gồm đèn LED (chân 1 , 2) và photo transistor ( chân 3,4).  Giữa hai thành phần này có điện áp cách ly với nhau vô cùng lớn lên tới 5kV

##### **Nguyên lý hoạt động**

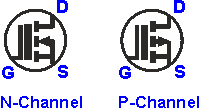
 một tín hiệu số được gửi đến đền LED thông qua một điện trở 330Ω. Phía bên photo transistor được mắc với từ chân C với nguồn 12V. Tín hiệu đầu ra sẽ lấy tại chân C của photo transistor. Khi tín hiệu đầu vào ở mức 0V thì đèn LED không có dòng điện chạy qua lên không sáng, photo transistor không dẫn điện và có tổng trở rất lớn, tín hiệu điện áp rơi trên hết trên chân C của photo transistor là 12V. Khi tín hiệu đầu vào ở mức 5V thì đèn LED sẽ có dòng điện chạy qua lên sẽ sáng, ánh sáng này chiếu vào photo transistor làm nó dẫn điện , trở kháng lúc này của photo transistor rất bé và nó gần như dẫn thông xuống mass làm tín hiệu điện trên chân C sụt về 0V.

Như vậy có thể thấy có thể dùng opto  để một tín hiệu 5V điều khiển một tín hiệu 12V thông qua ánh sáng.

1. **MOSFET ( IRF3205)**

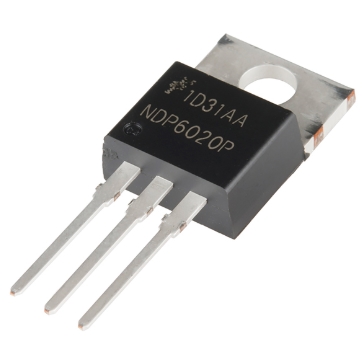


MOSFET IRF3205



##### **Định nghĩa:**

MOSFET (viết tắt tiếng Anh của Metal Oxide Semiconductor Field Effect Transistor) là thiết bị bán dẫn được sử dụng rộng rãi. Chúng ta có thể gặp nó ở rất nhiều trong những board nguồn xung, mạch điều khiển băm xung,… MOSFET có 2 loại là kênh N và kênh P. 3 chân MOSFET là chân G ( cực điều khiển), chân D (cực máng) và chân S ( cực gốc). 2 chân D và S được coi như là 2 chân tiếp tiếp công tắc và chân G là nút nhấn công tắc ấy. Nó được sử dụng như một công tắc điện tử, chế độ khuếch đại giống như transistor. Tuy nhiên MOSFET thì được ứng dụng nhiều hơn trong các mạch đóng cắt mạch điện tử bởi đặc tính đóng cắt với tần số cao hơn so với transistor.

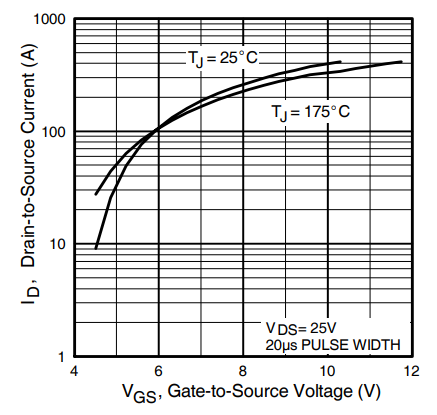


Hình 25. Hình ảnh thực tế của MOSFET

##### **Nguyên lí hoạt động:**

Chân G của Mosfet sẽ được nối với chân C của photo transistor. Chân D sẽ được nối với một đầu của động cơ, đầu còn lại được nối với nguồn 12V. Chân S được nối xuống mass.

Mosfet là linh kiện được điều khiển bằng điện áp cấp vào chân G. Khi điên áp điều khiển nhỏ hơn một ngưỡng nào đó, cỡ 3V, Mosfet ở trạng thái khóa và điện trở rất lớn giữa cực máng D và cực gốc S. Khi cỡ 5 – 7V, Mosfet sẽ ở chế độ dẫn. Thông thường người ta điều khiển Mosfet bằng điện áp điều khiển cỡ lớn hơn 10V để làm giảm điện áp rơi trên D và S và Mosfet sẽ dẫn tối đa. Khi đó dòng là dòng qua tải sẽ tỷ lệ với điện áp kích

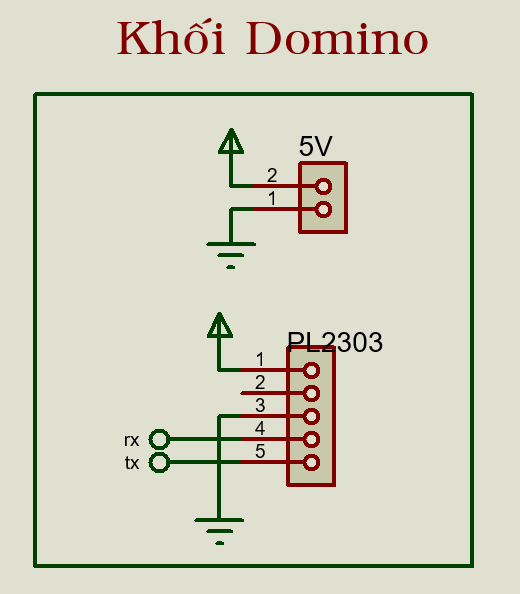


#### **Động cơ quạt DC 12V**

****

Hình 26. Quạt DC 12v thực tế

### **Khối giao tiếp với mạch tính và kết nối với mạch nguồn**

****

Hình 27. Khối Giao tiếp máy tính

[**Mạch Chuyển USB UART PL2303**](https://caka.vn/)sử dụng chip [**PL2303HX**](https://caka.vn/) chuyển đổi USB – UART dễ dàng kết nối với máy tính. Module dễ dàng cho việc nghiên cứu các module khác bằng cách gửi lệnh trực tiếp từ máy tính và phân tích dữ liệu nhận được lên màn hình máy tính mà không cần thông qua chương trình của vi điều khiền.

Thông số kỹ thuật

* Điện áp 5V cấp trực tiếp từ cổng USB.
* Ngõ ra dạng USART gồm 2 chân TX, RX.
* Với 3 led trên board: led báo nguồn, led RX, led TX.
* Kích thước: 15 x 31 mm.​



Hình 28. Ảnh thực tế của USB UART PL2303

**Giao tiếp cổng COM với chuẩn rs232**

Chúng tôi sử dụng chuẩn giao tiếp RS232 để giao tiếp vi điều khiển với máy tính quan sát thông số và chế độ hoạt động của mạch. Nó là một chuẩn giao tiếp nối tiếp dùng định dạng không đồng bộ, kết nối nhiều nhất là 2 thiết bị, chiều dài kết nối lớn nhất để đảm bảo dữ liệu là 25m, tốc độ 20kbit/s.

* Ưu điểm của giao tiếp nối tiếp RS232:

+ Khả năng chống nhiễu của các cổng nối tiếp cao.

+ Thiết bị ngoại vi có thể tháo lắp ngay cả khi máy tính đang được cấp điện.

+ Các mạch điện đơn giản có thể nhận được điện áp nguồn nuôi qua cổng nối tiếp

Các máy tính thường có một hoặc hai cổng nối tiếp theo chuẩn RS232 được gọi là cổng COM. Tuy nhiên trong các dòng laptop hiện đại ngày nay thì các cổng COM không còn được tích hợp trong main của máy tính. Để sử dụng cổng COM đối với loại laptop này ta cần phải có các thiết bị chuyển đổi tương tự như cáp chuyển đổi USB to COM. Ở đây, chúng tôi sử dụng mạch chuyển đổi USB UART PL2303.

**Phần mềm giao tiếp cổng com**

Giao tiếp chuẩn RS232 và vi điều khiển phải thông qua phần mềm giao diện để nhận biết được thông số truyền lên và chế độ hoặt động của mạch. Chúng tôi tạo một giao diện giao tiếp giữa máy tính với vi điều khiển thông qua GUI trên Matlab. Matlab có hỗ trợ truyền thông với ngoại vi thông qua

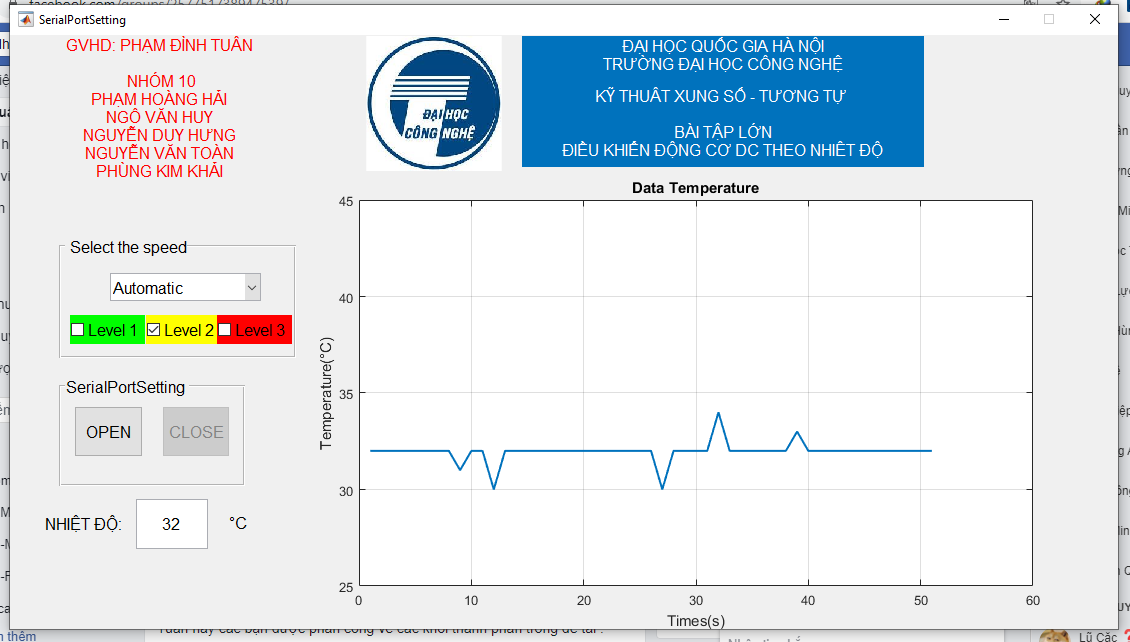
cổng Serial. Khi đó chúng ta dù không có mặt ở phòng đo nhiệt dộ nhưng

vẫn biết được nhiệt độ phòng và tốc độ của động cơ thông qua chương trình

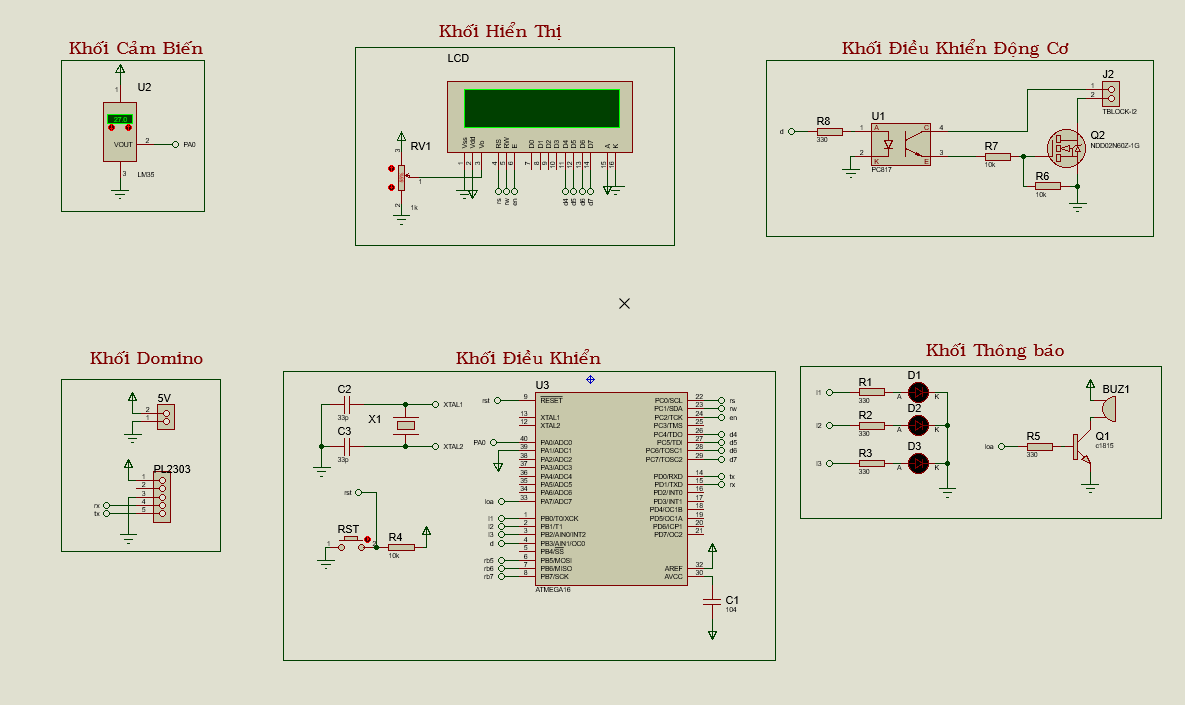
trên máy tính.

Một đồ thị biểu đồ nhiệt độ theo thời gian sẽ giúp người dùng dễ dàng đánh

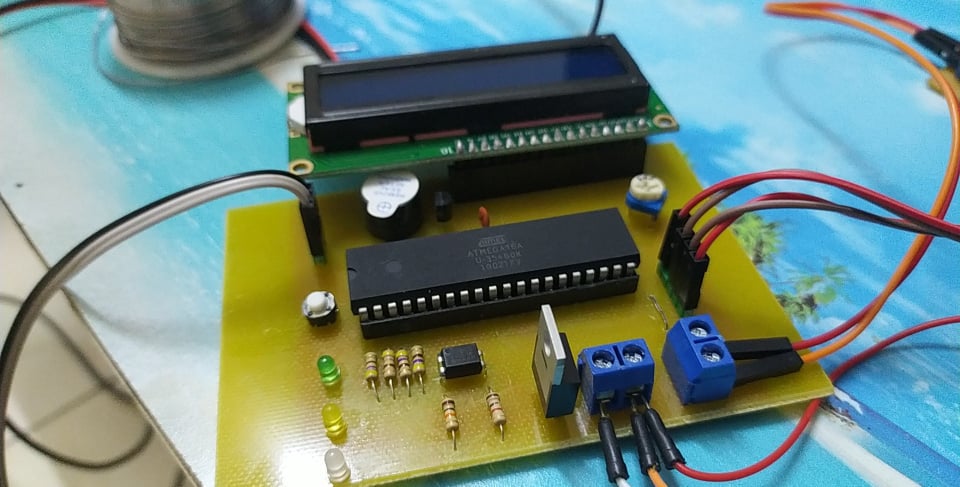
giá, phân tích được dữ liệu nhiêt độ gửi về.



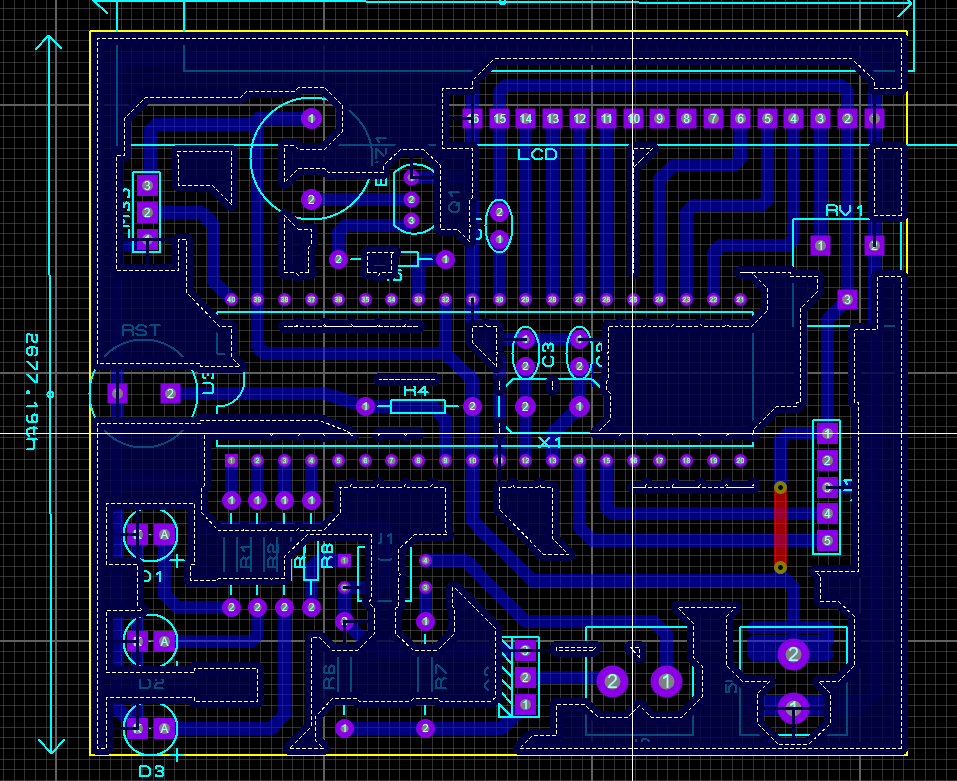
**Sơ đồ mạch hoàn chỉnh**

****

Hình 29. Sơ đồ mạch đo lường và điều khiển hoàn chỉnh

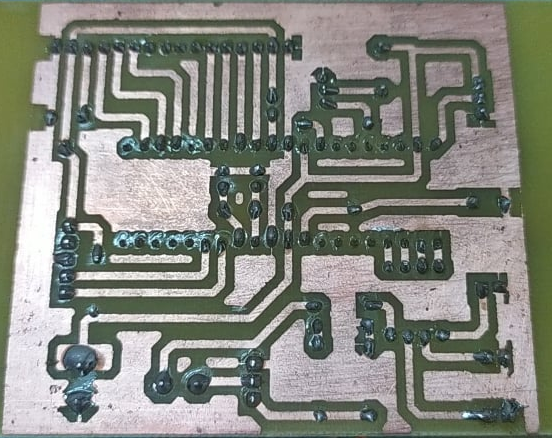


Hình 30. Ảnh mạch thực tế

****

Hình 31. Ảnh mạch đi dây PCB bằng Proteus

Do lần đầu hàn mạch in nên nhóm em còn nhiều thiếu sót phải chỉnh sửa nhiều lần mạch mới hoàn thiện.



Hình 32. Ảnh mạch di dây

**CHƯƠNG 2: THIẾT KẾ PHẦN MỀM**

Để lập trình cho vi điều khiển ta có thể dùng nhiều chương trình và ngôn ngữ lập trình khác nhau. Trong đồ án này, nhóm em sử dụng phần mềm Atmel Studio và viết chương trình bằng ngôn ngữ C.

1. **Thuật toán**

Nhóm em xây dựng chương trình dựa trên các nguyên tắc cơ bản như sau:

* Đo nhiệt độ phòng và hiển thị lên LCD 16x2

+ Sử dụng chân PA0(ADC0) đọc nhiệt độ bằng cảm biến LM35 hiển thị lên LCD.

* Sử dụng nhiệt độ đo được để thiết lập tốc độ quay cho động cơ.

+ Khi nhiệt độ đo được nhỏ hơn 25 thì động cơ không quay.

*+* Khi nhiệt độ đo được lớn hơn 25 và nhỏ hơn 30 độ, động cơ quay ở mức 1, độ rộng xung là 30%, đèn thông báo cấp độ 1 sáng.

+ Khi nhiệt độ đo được lớn hơn 30 độ và nhỏ hơn 35, động cơ quay ở mức 2, độ rộng xung là 70%, đèn thông báo cấp độ 2 sáng.

+ Khi nhiệt độ đo được lớn hơn 35 độ, động cơ quay ở mức 3, độ rộng xung là 100%, đèn thông báo cấp độ 3 sáng, còi kêu để cảnh báo.

* Cứ mỗi 100ms vi điều khiển lại gửi dữ liệu lên máy tính

+ Matlab sẽ nhận được giá trị nhiệt độ tức thời sau đó vẽ biểu đồ nhiệt độ theo thời gian.

+ Matlab nhận được cấp độ quay của động cơ và hiển thị tương ứng trên màn hình.

1. **Code hoàn chỉnh**

|  |
| --- |
| #define F\_CPU 8000000UL  #include <avr/io.h>  #include <util/delay.h>  #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <avr/interrupt.h>  #include <avr/sfr\_defs.h>  #define LCD\_Dir DDRC //Xac dinh cong giao tiep LCD  #define LCD\_Port PORTC //Xac dinh cong du lieu LCD  #define RS PINC0 //Dinh nghia kenh RS  #define RW PINC1 //Ding nghia kenh EN  #define EN PINC2  #define degree\_sysmbol 0xDF //Bieu duong do  #define cbi(port, bit) port &= ~(1<<bit)  #define sbi(port, bit) port |= (1<<bit)  char Temperature[20];  float celsius;  char tx\_message[10];  void LCD\_Command( unsigned char cmnd )  {  LCD\_Port = (LCD\_Port & 0x0F) | (cmnd & 0xF0); //gui nibble cao  LCD\_Port &= ~ (1<<RS); //RS = 0 su dung thanh ghi IR  LCD\_Port &= ~ (1<<RW);  LCD\_Port |= (1<<EN); //EN = 1 Kich hoat xung  \_delay\_us(1);  LCD\_Port &= ~ (1<<EN);  \_delay\_us(200);  LCD\_Port = (LCD\_Port & 0x0F) | (cmnd << 4); //Gui nibble thap  LCD\_Port &= ~ (1<<RW);  LCD\_Port |= (1<<EN); //EN = 1 Kich hoat xung  \_delay\_us(1);  LCD\_Port &= ~ (1<<EN);  \_delay\_ms(2);  }  void LCD\_Char( unsigned char data )  {  LCD\_Port = (LCD\_Port & 0x0F) | (data & 0xF0); //Gui nibble cao  LCD\_Port |= (1<<RS);  LCD\_Port &= ~ (1<<RW); //Rs = 1 : thanh ghi DR  LCD\_Port|= (1<<EN);  \_delay\_us(1);  LCD\_Port &= ~ (1<<EN);  \_delay\_us(200);  LCD\_Port = (LCD\_Port & 0x0F) | (data << 4); // Gui nibble thap  LCD\_Port &= ~ (1<<RW);  LCD\_Port |= (1<<EN);  \_delay\_us(1);  LCD\_Port &= ~ (1<<EN);  \_delay\_ms(2);  }  void LCD\_Init (void) //Ham khoi tao LCD  {  LCD\_Dir = 0xFF; //output  \_delay\_ms(20); //Tao tre bat LCD luon> 15ms  LCD\_Command(0x02); //khoi tao lcd 4 bit  LCD\_Command(0x28); //2 dong, ma tran 5 \* 7 o che ?o 4 bit  LCD\_Command(0x0c); //Bat man hinh, tat con tro  LCD\_Command(0x06); //Con tro tang, dich chuyen con tro sang phai  LCD\_Command(0x01); //Xoa man hinh  \_delay\_ms(2);  }  void LCD\_String (char \*str) //Ham gui chuoi  {  int i;  for(i=0;str[i]!=0;i++) //Gui tung ki tu cho den ki tu Null  {  LCD\_Char (str[i]);  }  }  void LCD\_String\_xy (char row, char pos, char \*str) // Ham gui chuoi toi vi tri xy  {  if (row == 0 && pos<16)  LCD\_Command((pos & 0x0F)|0x80); //lenh cua hang dau tien va vi tri bat buoc <16  else if (row == 1 && pos<16)  LCD\_Command((pos & 0x0F)|0xC0); //lenh cua hang thu hai va vi tri bat buoc <16  LCD\_String(str); //Goi ham  }  void LCD\_Clear()  {  LCD\_Command (0x01); //xoa man hinh  \_delay\_ms(2);  LCD\_Command (0x80); //dua con tro ve vi tri ban dau  }  void ADC\_Init()  {  ADMUX = 0x40; // Can le phai, dien ap tham chieu = Avcc,  //chon kenh dau vao chuyen doi ADC0  ADCSRA = 0x83; //Bat ADC, bo Chia /8  }  int ADC\_Read()  {  ADMUX = 0x40; // Can le phai, dien ap tham chieu = Avcc,  //chon kenh dau vao chuyen doi ADC0  ADCSRA |= 0x43; // ADCSC = 1, bat dau chuyen doi, gia tri chia tan = 8  while(!(ADCSRA & 0x10)); // Kiem tra chuyen doi ket thuc chua  ADCSRA |= 0x10; // Xoa co ngat bang cach ghi 1 toi bit ADIF    return ADCW;  }  void UART\_Init() //ham khoi tao UART  {  UBRRH=0;  UBRRL=51; //Cai dat toc do Baud la 9600 voi tan so 8Mhz  UCSRA=0x00; //Khoi tao su dung UART  UCSRC=(1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0); // Set thanh ghi UCSRC, khong chan le, 8 bit du lieu, 1 bit stop  UCSRB|=(1<<RXEN)|(1<<TXEN)|(1<<RXCIE); //Khai bao truyen du lieu, nhan du lieu va ngat khi nhan  }  void UART\_TxChar(unsigned char ch) //Ham truyen ky tu UART  {  while (! (UCSRA & (1<<UDRE))); //Cho cho den khi bit UDRE = 0  UDR = ch;  }  void UART\_TxString(char\* str) //Hien thi  {  for(int i = 0; str[i] != 0; i++)  {  UART\_TxChar(str[i]); //Xuat ra mang can truyen  }  }  unsigned char UART\_RxChar()  {  while(! (UCSRA & (1<<RXC)));  return UDR;  }  int main()  {  DDRA = 0x00;  DDRB = 0xFF;  DDRD = 0xFD;    TCCR0 |=(1<<WGM00)|(1<<COM01)|(1<<COM00)|(1<<CS00)|(1<<CS02); // su dung phase correct PWM, inverted PWM, bo chia 1024  TIMSK=0;  OCR0=255; //Duty cycle = 0  LCD\_Init(); //Khoi tao LCD  ADC\_Init(); //Khoi tao ADC  UART\_Init();  while(1)  {  LCD\_String\_xy(0, 0,"NHIET DO:");  celsius = (ADC\_Read()/2.046);  sprintf(Temperature,"%d%cC ", (int)celsius, degree\_sysmbol); //Chuyen doi gia tri so nguyen thanh chuoi ASCII  LCD\_String\_xy(0, 9,Temperature); //gui du lieu chuoi de in  \_delay\_ms(1000);  memset(Temperature,0,10); //sao chep chuoi de inxs  if(celsius < 25)  {  cbi(PORTB,0); // Tat den 0  cbi(PORTB,1); // Tat den 1  cbi(PORTB,2); // Tat den 2  OCR0=255; // khong cho quat quay  cbi(PORTA,7); // Buzzer tat  LCD\_String\_xy(1, 0,"NHIET DO ON DINH");  }  if(celsius >= 30 && celsius < 35)  {  sbi(PORTB,1); // Bat den 1  cbi(PORTB,0); // Tat den 0  cbi(PORTB,2); // Tat den 2  OCR0=76; // Cho quat quay o cap do 2  cbi(PORTA,7); // Buzzer tat  LCD\_String\_xy(1, 0,"CAP DO ");  LCD\_String\_xy(1, 7," 2: 70 % ");  }  if(celsius >= 35)  {  sbi(PORTB,2); // Bat den 2  cbi(PORTB,0); // Tat den 0  cbi(PORTB,1); // Tat den 1  OCR0=0; // Cho quat quay o cap do 3  sbi(PORTA,7); // Buzzer keu  LCD\_String\_xy(1, 0,"CAP DO ");  LCD\_String\_xy(1, 7," 3: 100 %");  }  sprintf(tx\_message, "%d\r\n", (int)celsius);  \_delay\_ms(0.1);  UART\_TxString(tx\_message);  }  } |

**CHƯƠNG 3: KẾT LUẬN, ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN**

* **Kết luận**

Sau hai tháng nghiên cứu và thực hiện đề tài ứng dụng vi điều khiển AVR điều khiển trong việc điều khiển tốc độ động cơ DC, chúng em đã hoàn thành tốt đề tài được giao, hoàn thiện mạch nguyên lý, sơ đồ mạch in và sản phẩm thực tế. Với nhiều khó khăn ban đầu khi mới bắt tay vào hoàn thiện sản phẩm với kiến thức ít ỏi của mình và kinh nghiệm làm mạch thực tế không có nhiều nên việc hoàn thiện đề tài gặp nhiều khó khăn.  
Với nỗ lực của các thành viên trong nhóm, chúng em đã hoàn thành sản phẩm đúng thời gian.

* **Đánh giá**
  1. **Ưu điểm**• Mạch hoạt động tốt, các khối trong mạch chạy ổn định, sự liên hệ giữa các khối chặt chẽ.  
     • Màn hình hiển thị tốt, rõ ràng, động cơ và khối khuếch đại hoạt động tốt, khối  
     nguồn ổn định tốt điện áp đầu ra.  
     • Mạch khá nhỏ gọn, chi phí sản xuất thấp.

• Tổn hao trên phần công suất nhỏ.  
• Đây là phương pháp đơn giản và thực thế.

* 1. **Nhược điểm**

• Mạch chỉ điều khiển được động cơ công suất nhỏ, không điều khiển được động cơ công suất lớn. Để điều khiển được động cơ công suất lớn thì cần phải có mạch nguồn khác.   
• Dải điều khiển hẹp.  
• Phương pháp này không được dùng khi yêu cầu độ chính xác cao về tốc độ và nhiệt độ.

• Do làm hoàn toàn bằng thủ công lên mạch khá mỏng manh, độ bền không cao.

* **Hướng phát triển:**

Để sản phẩm được hoàn chỉnh hơn nữa thì cần thiết kế mạch trên một board duy nhất để tăng thêm tính thống nhất và gọn nhẹ.

Muốn chạy được động cơ ăn dòng lớn hơn phải thiết kế lại bộ nguồn có khả năng cung cấp dòng ra lớn hơn. Sử dụng các mạch nguồn xung.

Phát triển thêm phần cứng và phần mềm có thể điều khiển được nhiều thiết bị hơn. Có thể thích hợp thêm các bộ điều khiển từ xa, giao tiếp không dây qua Bluetooth, sim hay internet sử dụng các các module chuyên dụng.

Lập trình thiết kế chương trình điều khiển trên điện thoại.

**TÀI LIỆU THAM KHẢO**

* Giáo Trình Vi Xử Lý Và Vi Điều Khiển – Nguyên Lý Ứng Dụng của thầy Phạm Mạnh Thắng(chủ biên), Hoàng Văn Mạnh.
* Phần mềm hỗ trợ : Proteus, Altium, Atmel Studio, MATLAB, Virtual Serial,...
* Trang web tham khảo:

<http://www.hocavr.com/>

<https://www.electronicwings.com/>