ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO BÀI TẬP LỚN

Môn học: Hệ thống vi xử lý

Đề tài: Điều khiển nhiệt độ ứng dụng trong lò ấp trứng.

Giảng viên: TS. Nguyễn Ngọc An.

Sinh viên: Ngô Minh Khánh – 18020698
Đặng Trung Hiếu – 18020498

Mục lục

I. Mở đầu.

1. Đặt vấn đề.

Đề tài đặt ra là thiết kế mạch điều khiển nhiệt độ ứng dụng trong lò ấp trứng sử dụng VĐK PIC16F877A là VĐK chính. Do đó mạch phải bao gồm các tính năng gia nhiệt khi nhiệt độ dưới ngưỡng cũng như khả năng thông gió, tản nhiệt khi nhiệt độ vượt ngưỡng. Bên cạnh đó, mạch cần có tính linh hoạt có thể điều chỉnh ngưỡng nhiệt độ mong muốn và dễ theo dõi, kiểm tra trong quá trình hoạt động.

Từ việc phân tích và khảo sát trên đây, nhóm đã đưa ra các vấn đề chính cần giải quyết của bài toán đặt ra là:

- Tìm hiểu kiến thức cơ bàn về VĐK PIC16F877A
- Xây dựng sơ đồ khối, từ đó xây dựng mạch nguyên lý, mô phỏng trên phần mềm Proteus.
- Thiết kế mạch in trên phần mềm Altium Designer / EasyEDA.
- Thực hiện lắp ráp linh kiện trên mạch in.
- Cài đặt và sử dụng IDE MPLABX với trình biên dịch XC8.
- Lắp đặt mô hình.
- Viết báo cáo tổng hợp về quy trình thực hiện đề tài.

2. Yêu cầu chức năng.

Các yêu cầu chức năng:

- Đo đạc nhiệt độ.
- Điều chỉnh nhiệt độ cài đặt.
- Hiện thị nhiệt độ cài đặt và nhiệt độ hiện tại.
- Điều chỉnh thiết bị gia nhiệt dùng thuật toán PID.
- Tính năng tản nhiệt.

3. Yêu cầu phi chức năng.

Các yêu cầu phi chức năng:

- Vi điều khiển: PIC16F877A.
- Thiết bị gia nhiệt: Bóng đèn 220V-60W.
- Thiết bị tản nhiệt: Quạt 12V DC.
- Nguồn hoạt động: 12V DC.
- Dễ dàng cấp nguồn với jack DC.

4. Nhiệt độ ấp đối với một số loại trứng thông dụng.

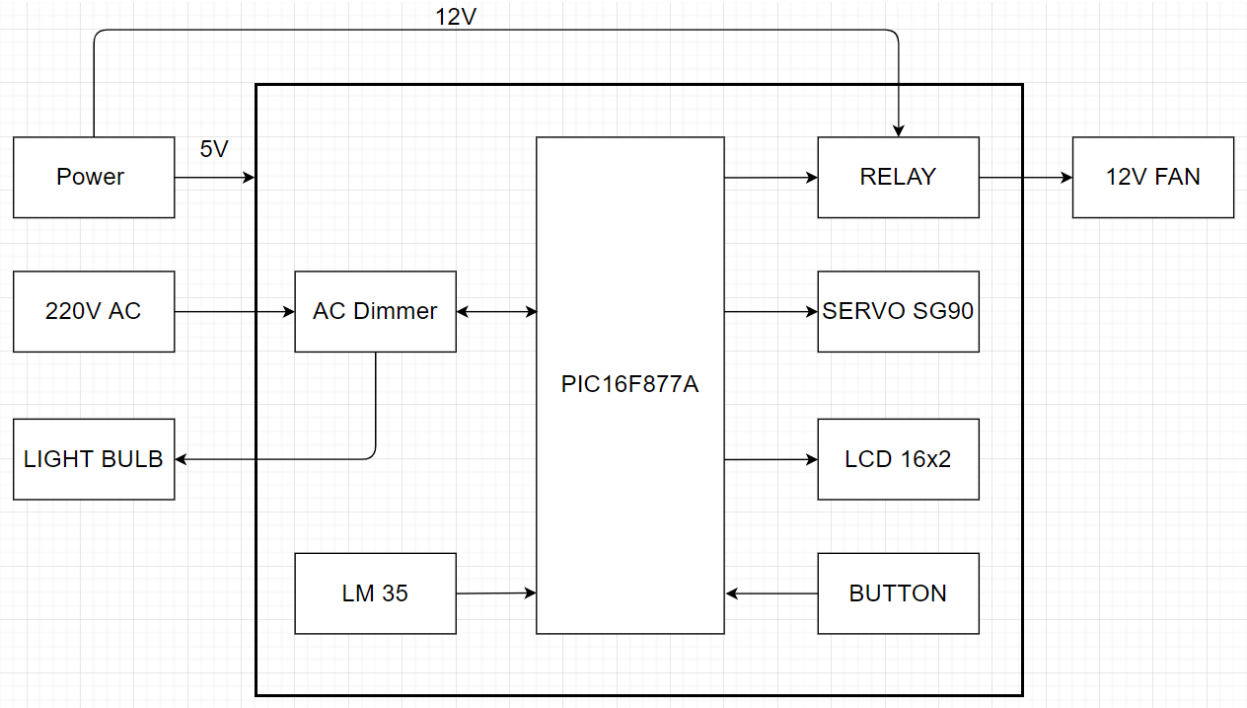
Dưới đây là bảng liệt kế nhiệt độ ấp phù hợp và thời gian ấp của một số loại trứng mà nhóm em đã tìm hiểu:

Loại trứng	Nhiệt độ (độ C)	Thời gian ấp (ngày)
Gà	37.4 – 37.8	21
Vịt	37.4 – 37.8	28
Ngan	37.3 – 37.8	35 – 37
Ngỗng	37.3 – 37.8	28 – 34
Gà lôi	37.4 – 37.8	23 – 28
Chim cút	37.0 – 37.2	17
Bồ câu	37.5	17

Bảng 1. Thống kê nhiệt độ và thời gian ấp phù hợp đối với 1 số loại trứng.

II. Thiết kế hệ thống.

1. Sơ đồ khối thành phần hệ thống.



Hình 1. Sơ đồ khối hệ thống

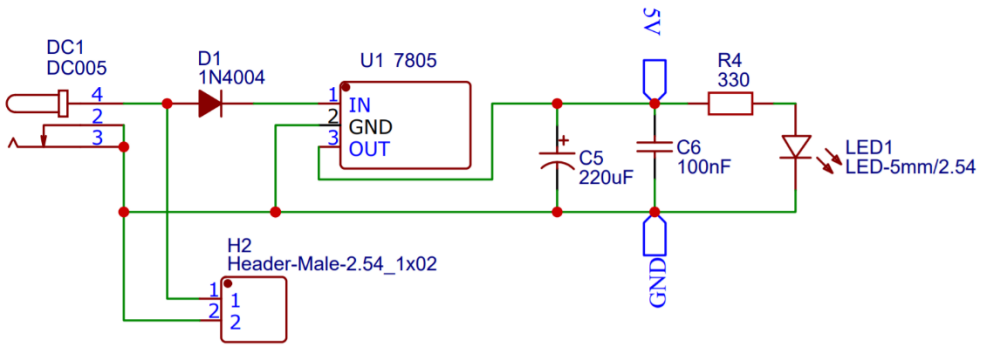
Hình 1 biểu diễn sơ đồ khối tổng quan của toàn bộ đề tài:

- Nguồn: Nguồn cung cấp đầu vào có điện áp 12V được hạ áp xuống 5V dùng 7805. Điện áp vào 12V cũng được cung cấp cho relay để bật/tắt quạt.
- PIC16F877A: Xử lý dữ liệu và đưa ra các tín hiệu điều khiển
- AC Dimmer: Được cấp nguồn 220V AC, mạch phát hiện điểm 0 gui tín hiệu ngắt về VĐK và nhận tín hiệu kích ra điều khiển bóng đèn.
- LM35: Đọc giá trị nhiệt độ gủi đến VĐK.
- Relay: Nhận tín hiệu điều khiển bật/tắt quạt.
- Servo SG90: Mở/đóng cửa thông gió.
- LCD1602: Hiện thị chỉ số nhiệt độ.

- Button: Điều chỉnh giá trị ngưỡng.

2. Hệ thống chi tiết.

a, Khối nguồn.

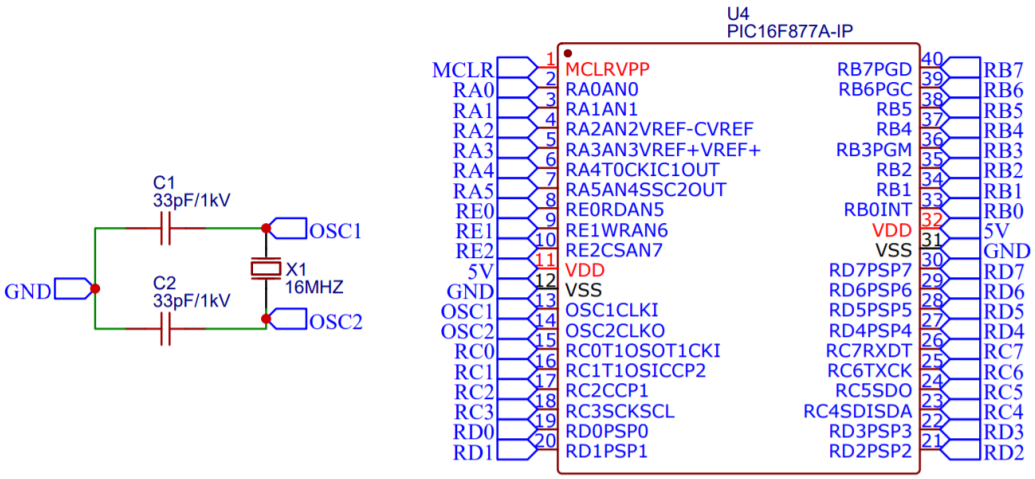


Hình 2.Mạch nguyên lý khối nguồn.

Khối nguồn sử dụng IC ổn áp 7805 để chuyển đổi điện áp vào 12V thành điện áp 5V cho các khối phía sau hoạt động. Diode D1 có vai trò chống dòng ngược. Tụ C5 có vai trò cung cấp điện áp tạm thời khi nguồn điện vào vị sụp giảm. Tụ C6 có vai trò lọc phẳng tín hiệu điện áp ra.

Khối có một Header đực 2.54mm để làm nguồn cung cấp nguồn 12V cho quạt.

b, Khối xử lý (PIC16F877A).

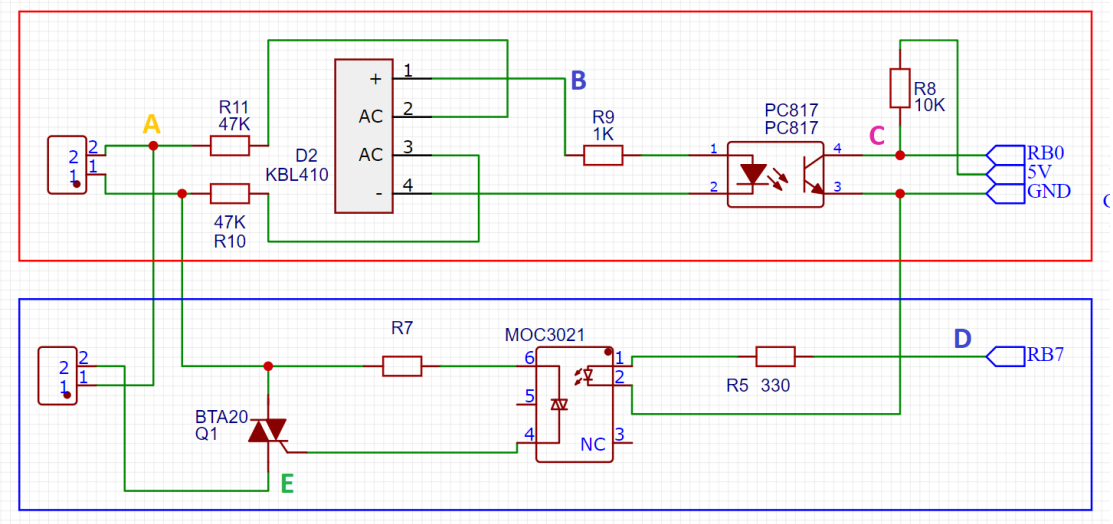


Hình 3. Mạch nguyên lý PIC16F877A.

Giới thiệu về PIC16F877A:

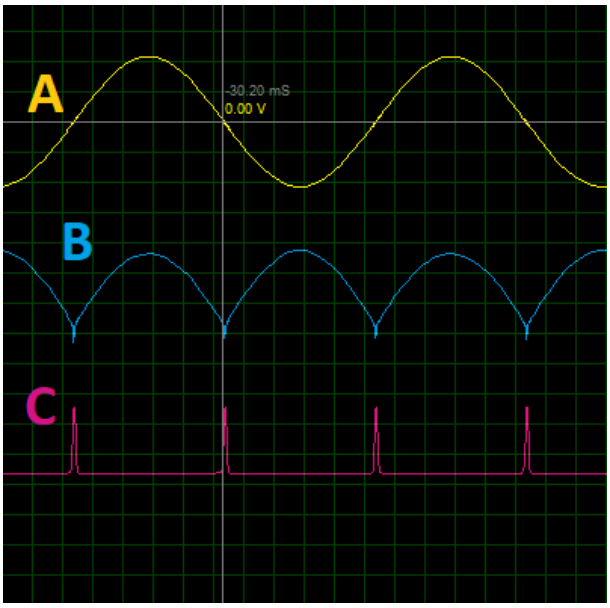
- 8K x 14 bits/word Flash ROM.
- 368 x 8 Bytes RAM.
- 256 x 8 Bytes EEPROM.
- 5 Port xuất/nhập (A, B, C, D, E) tương ứng 33 chân ra.
- 2 Bộ định thời 8 bit Timer 0 và Timer 2.
- 1 Bộ định thời 16 bit Timer 1, có thể hoạt động ở chế độ tiết kiệm năng lượng (SLEEP MODE) với nguồn xung clock ngoài.
- 2 Bộ Capture/ Compare/ PWM.(Bắt Giữ/ So Sánh/ Điều Biến Xung)
- 1 Bộ biến đổi Analog to Digital 10 bit, 8 ngõ vào.
- 2 Bộ so sánh tương tự (Compartor).
- 1 Bộ định thời giám sát (Watch Dog Timer).
- 1 Cổng giao tiếp song song 8 bit.
- 1 Port nối tiếp.
- 15 Nguồn ngắt (Interrupt).
- Chế độ tiết kiệm năng lượng (Sleep Mode).
- Nạp trương trình bằng cổng nối tiếp (ICSP™)(In-Circuit Serial Programming™ -)
- Tập lệnh gồm 35 lệnh có độ dài 14 bit.
- Tần số hoạt động tối đa 20 MHz.

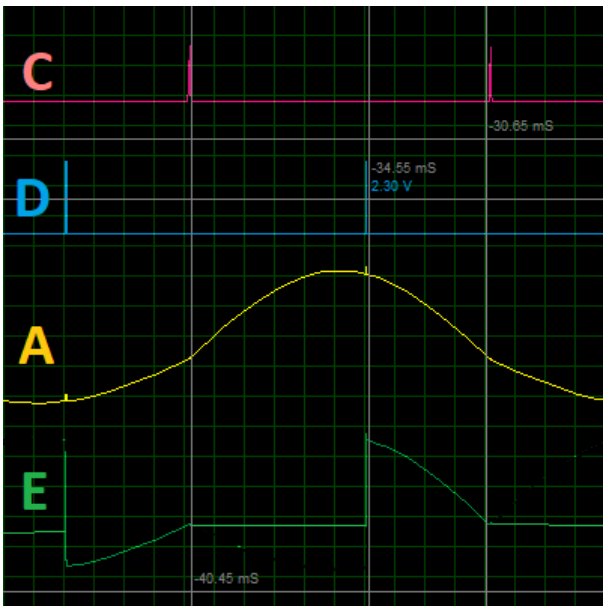
c, AC Dimmer.



Hình 4. Mạch AC dimmer

Mạch dimmer gồm hai thành phần chính: mạch phát hiện điểm 0 (khung đỏ) vàmạch điều khiển Thyristor (khung xanh).





Hình 5.a. *Hình 5.b.*

5.a. Giản đồ tín hiệu của mạch phát hiện điểm 0.

5.b. Giản đồ tín hiệu của mạch điều khiển Thyristor.

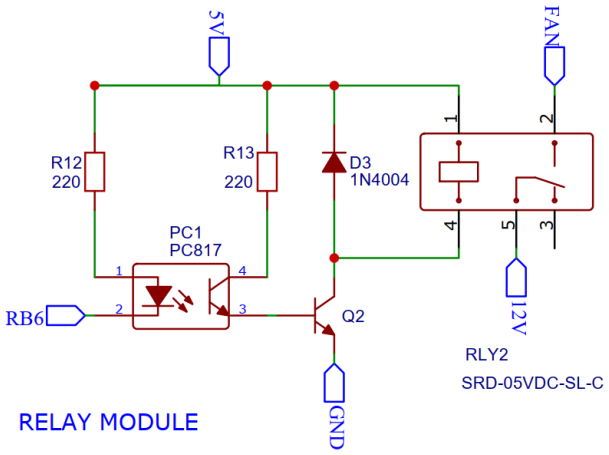
- Mạch phát hiện điểm 0: Phát hiện điểm 0 của tín hiệu xoay chiều, làm ngắt cho VĐK

Điện áp xoay chiều dạng sin tại điểm A. Điện áp xoay chiều sau đó được chỉnh lưu bằng cầu diode KBL410 đến điểm B. Tại đây ta dùng một opto PC817 để cách ly quang tín hiệu chỉnh lưu và VĐK. Khi điện áp tại B về (< 0.7), diode quang cấm khiến transistor quang cấm, điện áp tại C là 5V. Khi điện áp tại B > 0.7, diode quang dẫn, khiến transistor quang dẫn, điện áp tại C là 0V. (Hình 5.a)

- Mạch điều khiển Thyristor: điều khiển cường độ của bóng đèn.

Khi D có tín hiệu xung kích từ vi điều khiển tới opto MOC3021, làm triac quang bên trong opto dẫn, làm một tín hiệu điện áp xoay chiều qua R7 đến cực gate của thyristor khiến thyristor dẫn. Việc đóng ngắt thyristor đóng vai trò như là PWM trong điện xoay chiều. Tuy nhiên điện xoay chiều có dạng sóng sine do đó giải pháp là đóng ngắt theo nửa chu kỳ. Để điều khiển bóng đèn ta cần phối hợp với mạch phát hiện điểm 0. Ta gọi khoảng thời gian T là khoảng thời gian có tín hiệu ngắt từ mạch phát hiện điểm 0 đến khi có xung kích thyristor, T càng lớn thì cường độ bóng đèn càng thấp và ngược lại. (Hình 5.b)

d, Relay



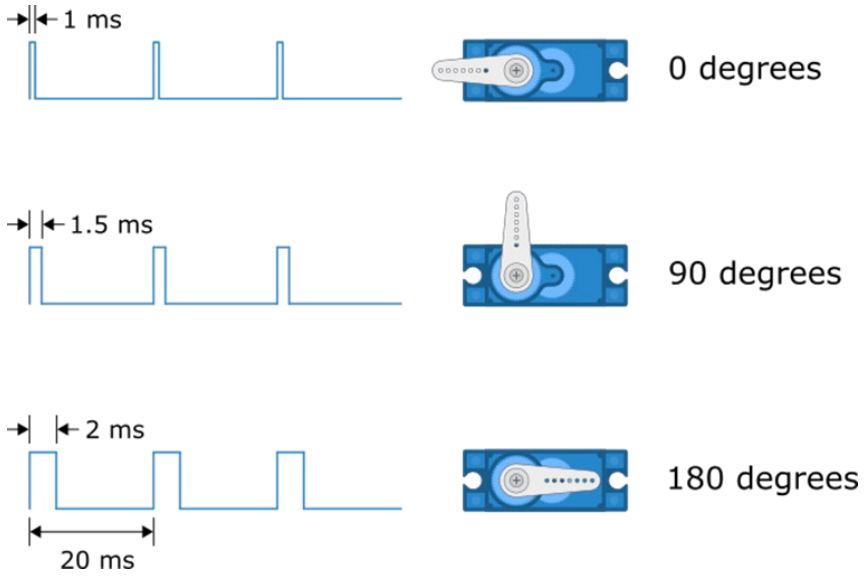
Hình 6. Mạch nguyên lý relay.

Khi RB6 bằng 0, diode quang trong PC817 thông khiến transistor quang dẫn. Q2 dẫn. Dòng điện đi qua cuộn dây sinh ra từ trường hút tiếp khóa từ tiếp điểm (NC) về tiếp điểm 2 (NO), nối quạt với nguồn 12V.

PC817 có vai trò cách ly quang, tránh nhiễu từ cho các thiết bị phía trước.

Diode D3 được mắc ngược song song với cuộn dây để bảo vệ các linh kiện trong mạch relay bằng cách dập các xung điện áp khi relay được ngắt.

e, Servo.



Hình 7. Nguyên lý điều khiển servo

Servo SG90 dùng để mở cửa thông gió khi cần tản nhiệt.

Servo SG90 được điều khiển bằng phương pháp PWM với chu kỳ 20ms. Tùy thuộc vào duty cycle mà ta có thể điều chỉnh góc của servo.

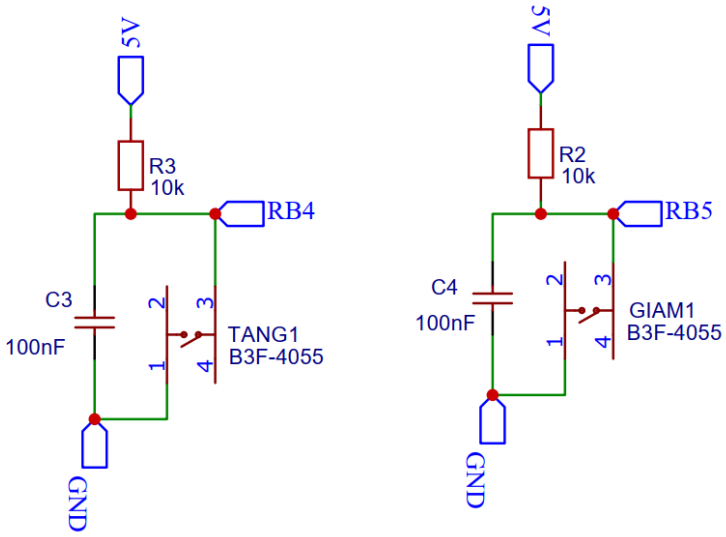
f, LM35.

Hình 8. Cảm biến nhiệt độ LM35.

LM35 là một loại cảm biến nhiệt độ bán dẫn tương tự rất phổ biến và được ứng dụng rất nhiều trong thực tế hiện nay. Nhiệt độ được xác định bằng cách đo hiệu điện thế ngõ ra và không cần căn chỉnh.

Nhiệt độ thay đổi tuyến tính: 10mV/°C.

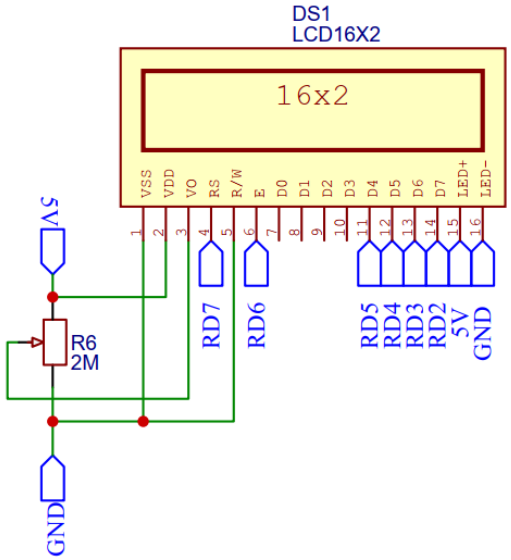
g, Khối nút nhấn.



Hình 9. Mạch nguyên lý nút nhấn.

Khối gồm 2 nút nhấn đề điều chỉnh tăng, giảm nhiệt độ cài đặt, tín hiệu ngắt cho vi điều khiển ở mức logic 0. Tụ C3 và C4 đóng vai trò chống dội phím bằng phần cứng.

h, Khối hiển thị (LCD1602).



Hình 10. Mạch nguyên lý hiện thị LCD.

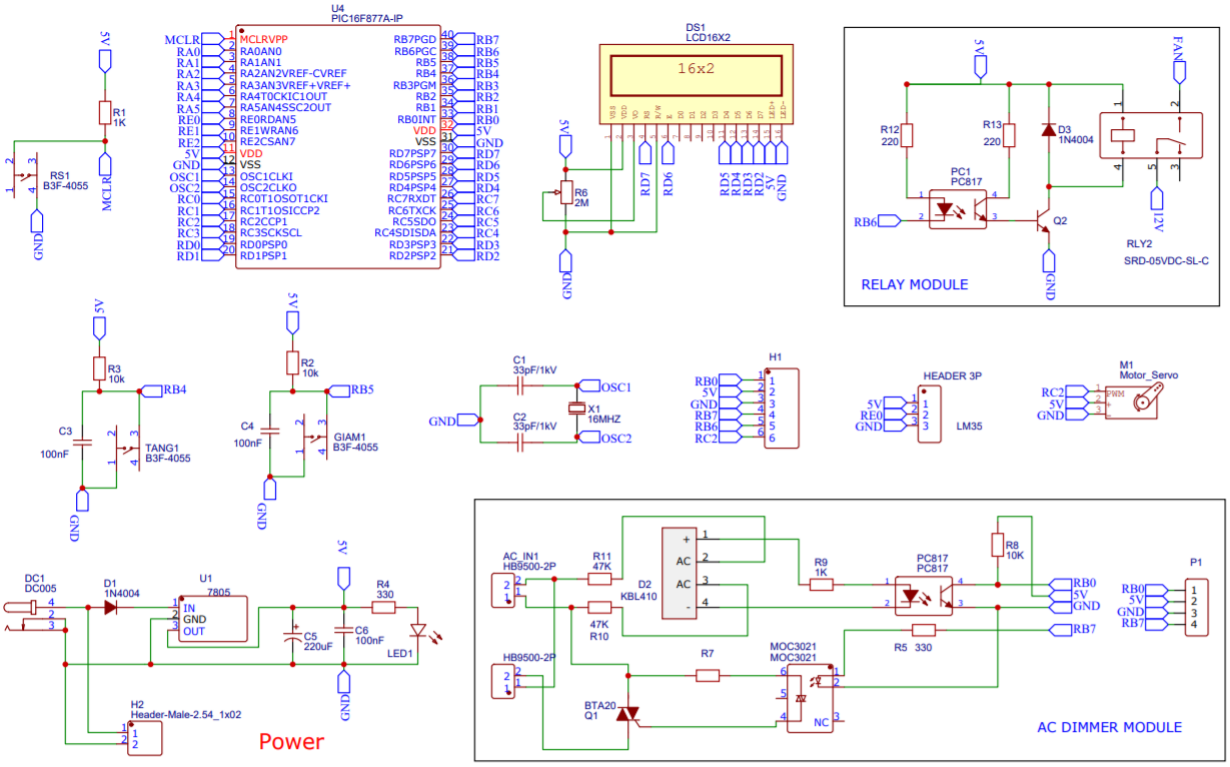
Khi sản xuất LCD, nhà sản xuất đã tích hợp chíp điều khiển (HD44780) bên trong lớp vỏ và chỉ đưa các chân giao tiếp cần thiết. Các chân này được đánh số thứ tự và đặt tên như bảng sau:

STT	Ký hiệu	Mô tả
1	VSS	Chân nối đất cho LCD, khi thiết kế mạch ta nối chân này với GND của mạch điều khiển
2	VDD	Chân cấp nguồn cho LCD, khi thiết kế mạch ta nối chân này với VCC=5V của mạch điều khiển
3	V0/VEE	Điều chỉnh độ tương phản của LCD.
4	RS	Chân chọn thanh ghi (Register select). Nối chân RS với logic "0" (GND) hoặc logic "1" (VCC) để chọn thanh ghi. + Logic "0": Bus DB0-DB7 sẽ nối với thanh ghi lệnh IR của LCD (ở chế độ "ghi" - write) hoặc nối với bộ đếm địa chỉ của LCD (ở chế độ "đọc" - read) + Logic "1": Bus DB0-DB7 sẽ nối với thanh ghi dữ liệu DR bên trong LCD.
5	R/W	Chân chọn chế độ đọc/ghi (Read/Write). Nối chân R/W với logic "0" để LCD hoạt động ở chế độ ghi, hoặc nối với logic "1" để LCD ở chế độ đọc.

6	E	Chân cho phép (Enable). Sau khi các tín hiệu được đặt lên bus DB0-DB7, các lệnh chỉ được chấp nhận khi có 1 xung cho phép của chân E. + Ở chế độ ghi: Dữ liệu ở bus sẽ được LCD chuyển vào(chấp nhận) thanh ghi bên trong nó khi phát hiện một xung (high-to-low transition) của tín hiệu chân E. + Ở chế độ đọc: Dữ liệu sẽ được LCD xuất ra DB0-DB7 khi phát hiện cạnh lên (low-to-high transition) ở chân E và được LCD giữ ở bus đến khi nào chân E xuống mức thấp.
7 - 14	DB0 - DB7	Tám đường của bus dữ liệu dùng để trao đổi thông tin với MPU. Có 2 chế độ sử dụng 8 đường bus này : + Chế độ 8 bit : Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7. + Chế độ 4 bit : Dữ liệu được truyền trên 4 đường từ DB4 tới DB7, bit MSB là DB7
15	Led+/ A	Nguồn dương cho đèn nền
15	Led-/K	GND cho đèn nền

Bảng 2. Chức năng các chân kết nối LCD1602.

3. Mạch nguyên lý toàn mạch.

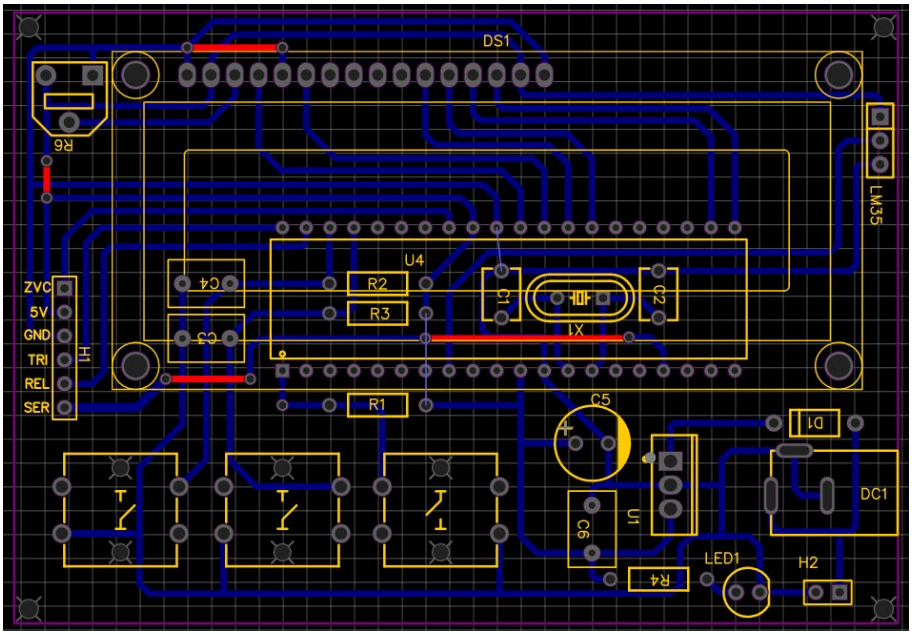


Hình 11. Sơ đồ nguyên lý toàn mạch

Để tận dụng những module relay đã có sẵn và tái sử dụng module dimmer vào những mục đích tương tự với các dòng vi điều khác khác, nhóm em chia mạch nguyên lý thành 3 phần chính: AC Dimmer module, Relay module và PIC16F877A_LM35_LCD (phần còn lại).

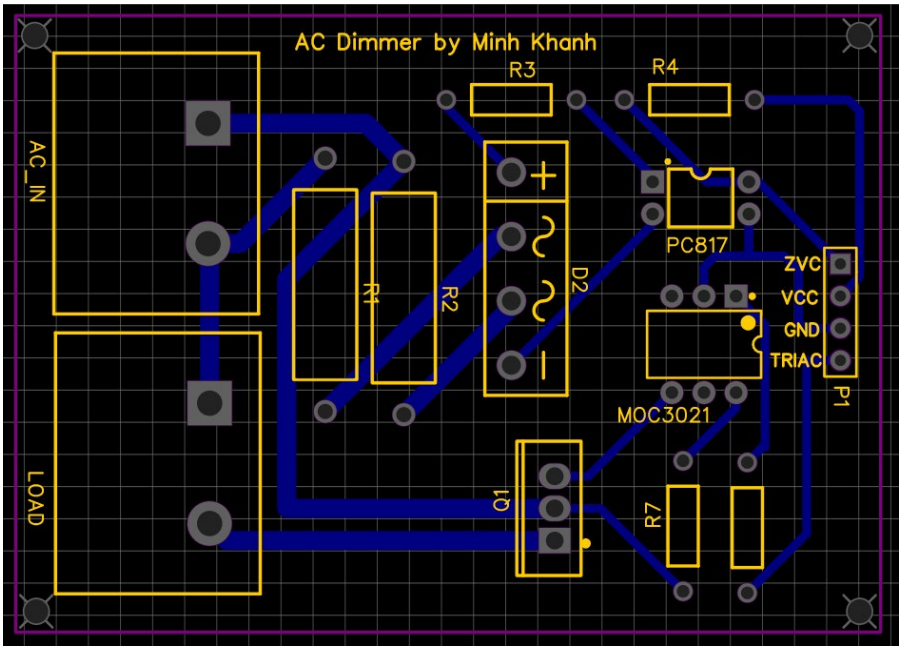
4. Thiết kế mạch in.

a, PIC16F877A_LM35_LCD



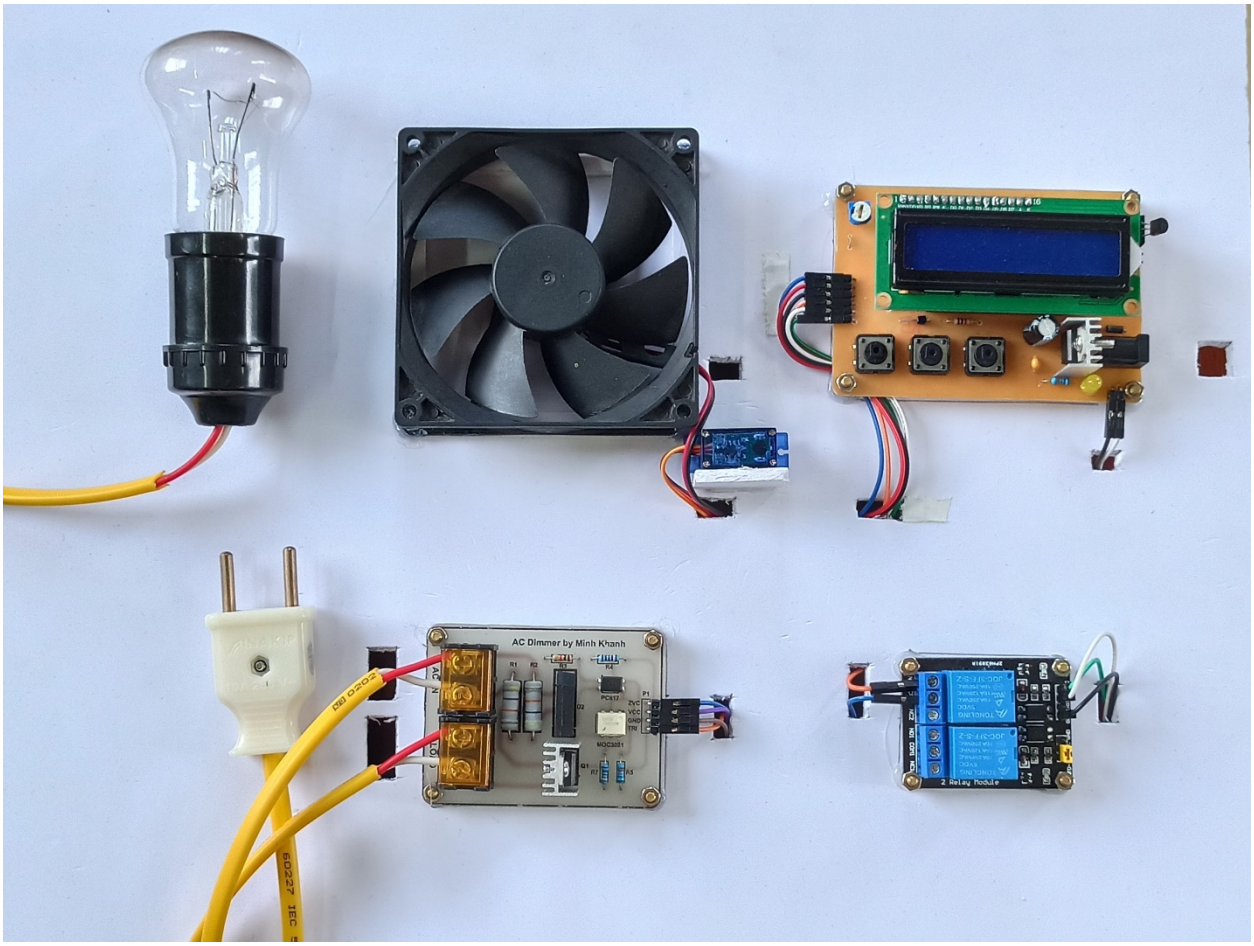
Hình 12. Mạch in PIC16F877A_LM35_LCD

b, AC Dimmer Module



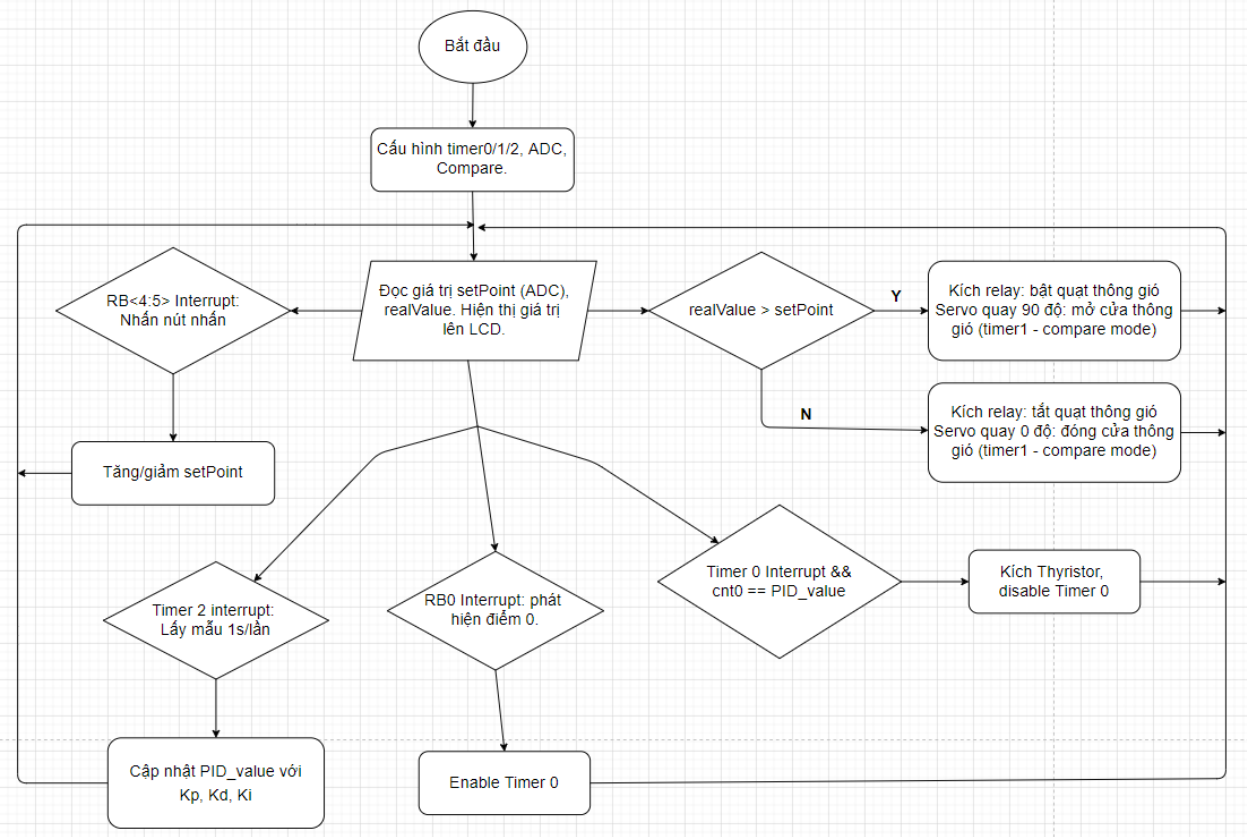
Hình 13. Mạch in AC Dimmer

5. Ảnh thực tế.



Hình 14. Ảnh thực tế.

III. Chương trình.



Hình 14. Sơ đồ thuật toán tổng quát.

Dưới đây nhóm em giải thích code của ADC và các điều khiển chính là: điều khiển bóng đèn bằng PID , Servo.

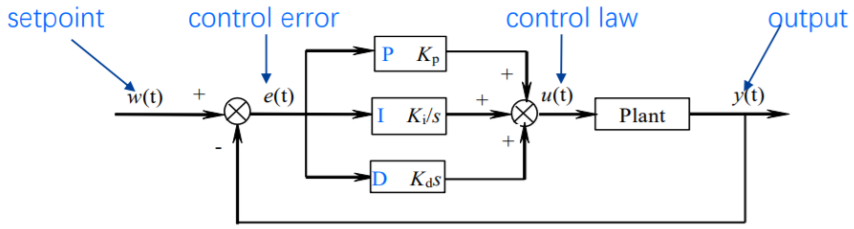
1. Thuật toán điều khiển PID.

a, Cơ sở lý thuyết.

Hệ thống điều khiển vòng kín là hệ thống sẽ xác định sai khác giữa trạng thái mong muốn và trạng thái thực (sai số) và tạo ra lệnh điều khiển để loại bỏ sai số. Điều khiển PID thực hiện ba cách phát hiện và hiệu chỉnh sai số này.

Hệ thống điều khiển có thể sử dụng P, PI, PD, hoặc PID để hiệu chỉnh sai số. Nhìn chung, vấn đề ở đây là "hiệu chỉnh" hệ thống bằng cách lựa chọn những giá trị thích hợp trong ba cách nêu trên.

- **Block diagram of a PID controller**



- **Textbook form**

$$e(t) = w(t) - y(t)$$

$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(s)\,\mathrm{d}s + K_{\mathrm{d}} \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

$$= K_{\mathrm{p}} \left(e(t) + \frac{1}{T_{\mathrm{i}}} \int_0^t e(s)\,\mathrm{d}s + T_{\mathrm{d}} \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right)$$

T_{i}: integration/reset time
T_{d}: derivative time

P I D

- **Digital PID controller**

$$u(t) = K_{\mathrm{p}} \left(e(t) + \frac{1}{T_{\mathrm{i}}} \int_0^t e(s)\,\mathrm{d}s + T_{\mathrm{d}} \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right)$$

  - – Numerical approximation for P, I, D terms

integral $\int$	summation, e.g., rectangle method $\sum$
derivative $\frac{\mathrm{d}}{\mathrm{d}t}$	backward/forward difference $\frac{(t) - (t-T)}{T}; \frac{(t+T) - (t)}{T}$

$$u_k = K_{\mathrm{p}} \left(e_k + \frac{T}{T_{\mathrm{i}}} \sum_{j=1}^{k} e_j + \frac{T_{\mathrm{d}}}{T} (e_k - e_{k-1}) \right)$$

**Trích từ slide của TS. Shanying Zhu – ĐH Giao thông Thượng Hải.*

b, Triển khai.

```
void Cal_PID(void)
{
    PID_error = setPoint - realValue;
    //integral constant only affect errors below 30
    if(PID_error > 30)
        PID_i = 0;
    //Calculate the P value
    PID_p = kp * PID_error;
    //Calculate the I value
    PID_i = PID_i + (ki * PID_error);
    //Calculate the D value
    PID_d = kd*(PID_error - previous_error);
    //Calculate total PID value
    PID_value = PID_p + PID_i + PID_d;
   //Store the previous error.
    previous_error = PID_error;
    if(PID_value < 1600)
        PID_value = 1600;
    if(PID_value > 8400)
        PID_value = 8400;
    PID_value = (10000 - PID_value);
    PID_value = PID_value / 100;
}
```

1. Tính lại giá trị PID sau mỗi 1s

```
if(INTF == 1)
{
    TMR0IE = 1;
    INTF = 0;
}
```

2. Enable Timer 0 khi phát hiện điểm 0.

```
if(TMR0IF == 1)
{
    TMR0 = 0x38;
    cnt0++;
    if(cnt0 == PID_value)
    {
        RB7 = 1;
        __delay_us(10);
        RB7 = 0;
        cnt0 = 0;
        TMR0IE = 0;
    }
    TMR0IF = 0;
}
```

3. Timer 0 được cài đặt interrupt mỗi 100us. Khi cnt0 đạt giá trị bằng PID_value, kích Thyristor và Disable Timer 0

Hình 15. Triển khai thuật toán PID.

***Chú thích:**

Mạch dimmer ta điều chỉnh đóng ngắt trong từng nửa chu kỳ của điện xoay chiều (Mục II.2.c). Điện xoay chiều có tần số 50Hz, nên nửa chu kỳ có thời gian = 0.01s = 10000us. Trong thực tế, khi khoảng thời gian giữa xung ngắt phát hiện điểm 0 và kích thyristor quá bé thì đèn sẽ không sáng, nên ta cần giới hạn khoảng thời gian bé nhất và khoảng thời gian lâu nhất kích thyristor kể từ khi có xung ngắt điểm 0.

Trong quá trình thực nghiệm dùng ngắt timer 0 để tạo thời gian đợi kích ở trên, timer 0 có khoảng mỗi lần ngắt bé nhất mà vẫn cho độ chính xác là 100us. Nên giá trị PID_value (khoảng thời gian đợi kích) được chia cho 100.

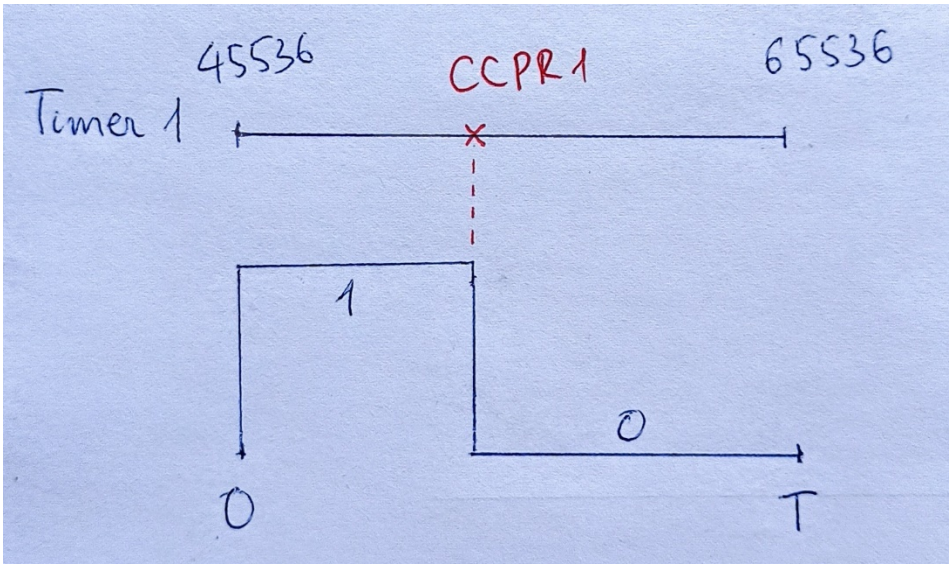
2. Điều khiển Servo.

$$PWM_{Period} = [(PR2) + 1] \times 4 \times T_{osc} \times Timer2PreScalerValue$$

Hình 16. Công thức tính chu kỳ PWM.

Với Tosc = 16MHz, PR2 (max) = 255, Timer2 prescale (max) = 16. Thì Tpwm = 1ms nhỏ hơn rất nhiều so với chu kỳ của SG90 = 20ms.

=> Dùng chế độ Compare, Timer 1 đóng vai trò chu kỳ và CCPR1 đóng vai trò điều chỉnh duty trong chu kỳ.



Hình 17. Mô tả PWM bằng chế độ Compare.

- Cài đặt chu kỳ bằng Timer 1:

  - Tần số thạch anh = 16MHz => Tần số PIC, f = 4MHz
  - Chọn prescale = 1:4
  - Chu kỳ cho 1 tick = 1/(f/4) = 1us

  ⇒ Cần tạo ra 20ms, ta cần 20000 tick

  ⇒ Đặt preload cho Timer 1 = 65536 – 20000 = 45536 (0xB1E6)

- Điều chỉnh Duty bằng CCPR1 (góc 0 và 90 độ):

  - Khi giá trị trên Timer 1 = CCP1R -> RC2 = 0.
  - Servo quay góc 0 độ: CPP1R = 0xB430.
  - Servo quay góc 90 độ: CCP1R = 0xB7A8.

```
// preload Timer1
TMR1 = 0xB1E0;
// Timer1 clock select bit
TMR1CS = 0;
// Prescaler ratio 1:4
T1CKPS0 = 0;
T1CKPS1 = 1;
// Switch ON Timer1
TMR1ON = 1;
```

1. Cấu hình Timer 1 Tạo chu kỳ 20ms.

```
TRISC2 = 0;
RC2 = 1;
// clear output on match.
CCP1CON = 0x09;
CCP1IE = 0;
CCP1IF = 0;
Servo_MoveTo(0);
```

2. Cấu hình cho compare mode. Xóa RC2 về 0 khi gí trị trên Timer 1 = CCPR1.

```
void Servo_MoveTo(int a)
{
    if(a==0)
    {
        CCPR1 = 0xB430;
    }
    if(a==90)
    {
        CCPR1 = 0xB7A8;
    }
}
```

3. Điều chỉnh duty cycle.

```
if(TMR1IF == 1)
{
    CCP1CON = 0x00;
    RC2 = 1;
    CCP1CON = 0x09;
    TMR1 = 0xB1E0;
    TMR1IF = 0;
}
```

4. Đặt RC2 lên 1, preload lại Timer 1, đặt lại chế độ compare khi kết thúc 1 chu kỳ (Timer 1 interrupt)

Hình 18. Code điều khiển Servo.

3. Đọc ADC.

- Cấu hình ADC:

```
void init_ADC (void)// adc
{
    // chon tan so clock cho bo adc
    ADCON1bits.ADCS2 = 0, ADCON0bits.ADCS1 = 1,ADCON0bits.ADCS0 = 0;
    // chon kenh adc la kenh AN5 - RE0
    ADCON0bits.CHS2 = 1, ADCON0bits.CHS1 = 0, ADCON0bits.CHS0 = 1;
    // chon cach luu data
    ADCON1bits.ADFM = 1;
    // cau hinh cong vao
    ADCON1bits.PCFG3 = 0,  ADCON1bits.PCFG2 = 0,  ADCON1bits.PCFG1 = 0,  ADCON1bits.PCFG0 = 0;
    // cap nguon cho khoi adc
    ADCON0bits.ADON = 1;
}
```

Hình 19. Code cấu hình ADC.

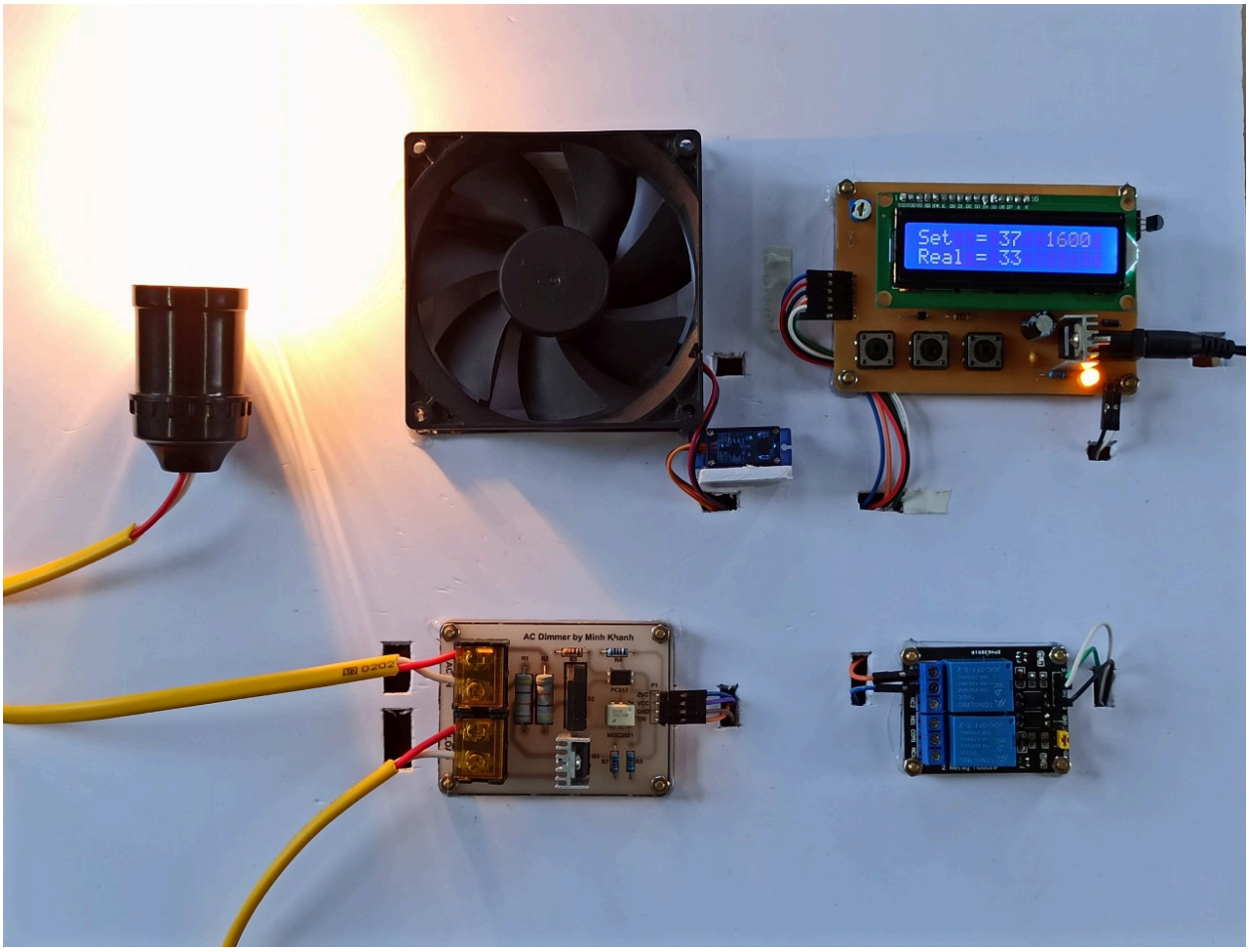
- Đọc giá trị nhiệt độ:

```
void Read_Temp (void)
{
    float TempValue = 0;
    ADCON0bits.GO_nDONE = 1;
    while(ADCON0bits.GO_nDONE);
    TempValue = ADRESH*256 + ADRESL;
    TempValue = 5000.0f / 1023 * TempValue;
    realValue = TempValue / 10;
}
```

Hình 20. Code đọc giá trị nhiệt độ

IV. Tổng kết.

1. Đánh giá hệ thống.

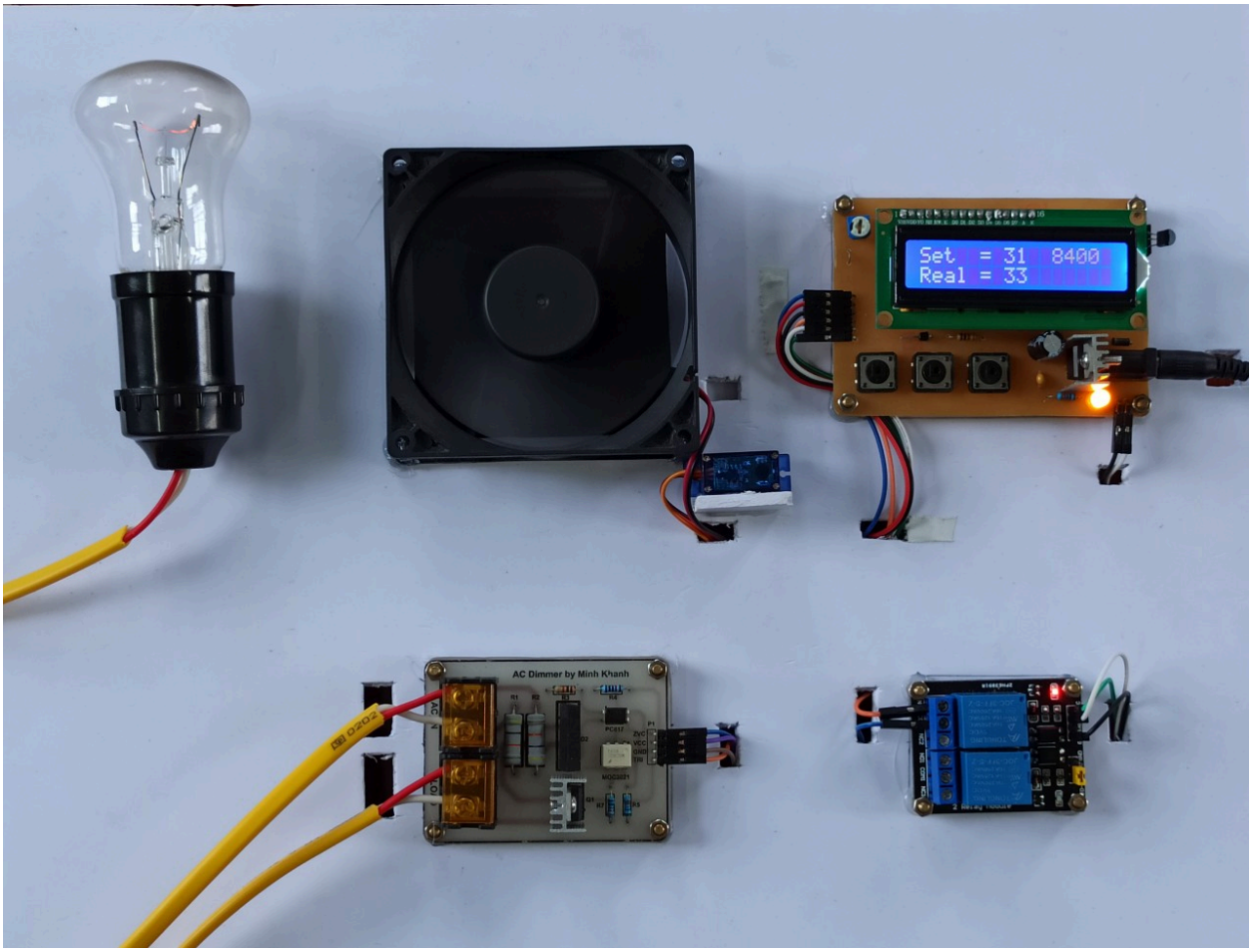


Hình 22. Hệ thống hoạt động với nhiệt độ thực thấp hơn nhiệt độ cài đặt.

- Khi nhiệt độ thực < nhiệt độ cài đặt:

- Vi điều khiển tính toán đưa ra khoảng thời gian để kích Thyristor từ khi phát hiện điểm 0 (trong ảnh khoảng thời gian là 1600us). Đèn sáng mạnh vì chưa đạt nhiệt độ đặt.
- Quạt tản nhiệt tắt.

- Cửa thông gió đóng.



Hình 23. Hệ thống hoạt động với nhiệt độ thực thấp hơn nhiệt độ cài đặt.

- Khi nhiệt độ thực > nhiệt độ cài đặt:

- Vi điều khiển tính toán đưa ra khoảng thời gian để kích Thyristor từ khi phát hiện điểm 0 (trong ảnh khoảng thời gian là 8400us). Đèn sáng yếu (gần như tắt) vì vượt nhiệt độ đặt.
- Quạt tản nhiệt bật.
- Cửa thông gió mở.

Hiện tại, nhóm em vừa thực hiện xong quá trình lắp đặt hệ thống, trong thời gian tới, chúng em sẽ thực hiện việc khảo sát hệ thống khi chạy thực tế và gửi đánh giá trong buổi báo cáo cuối kỳ.

2. Kết luận.

Qua việc thực hiện đề tài này chúng em đã hiểu và biết cách làm từng bước để thiết kế một hệ thống xử lý cụ thể công việc nào đó. Tuy nhiên với mục đích tìm hiểu ứng dụng của kỹ thuật vi xử lý và thời gian giới hạn, nhóm chúng em mới hoàn thành sản phẩm ở mức độ thử nghiệm. Trong quá trình thực hiện dự án này, chúng em đã học tập được thêm nhiều kiến thức thực tế, trao đổi thêm giữa các thành viên, làm quen với tác phong làm việc theo nhóm và cách thức xử lý các khó khăn khi gặp phải. Kỹ thuật vi xử lý là một môn học hay nhưng khó, đây cũng là lần đầu tiên nhóm thực hiện một dự án như thế này, do đó bên cạnh việc tự học thì sự hướng dẫn và các kinh nghiệm của thầy đã giúp đỡ chúng em rất nhiều. Chúng em cám ơn thầy Nguyễn Ngọc An và nhóm trợ giảng rất nhiều về những chỉ bảo tận tình trong thời gian qua!

Github: https://github.com/KhanhNgo19/HK_Egg_Incubator

V. Tài liệu tham khảo.

[1] Slide bài giảng thực hành Vi xử lý – Trường Đại học Công nghệ, ĐHQGHN.

[2] Datasheet vi xử lý PIC16F877A.

[3] Website https://deepbluembedded.com.

[4] https://github.com/yongxianan/TemperatureControllerMCU

[5] Slide bài giảng Computer Control Technology, TS. Shanying Zhu – ĐH Giao thông Thượng Hải.

[6] Nghiên cứu ba chế độ điều khiển On/Off, PID, Fuzzy trong điều khiển lò nhiệt – CN. Lê Tiến Lộc, CN. Lâm Thanh Hiển, Đại học Lạc Hồng.