ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA VẬT LÝ

BÁO CÁO BÀI TẬP CUỐI KÌ

Sinh viên: Trần Ngọc Huân 22000166
Trần Mai Anh 22000141
Lớp: K67 Vật lý học
Môn học: Vi điều khiển
Giảng viên hướng dẫn: PGS.TS. Lê Quang Thảo

Hà Nội – 2025

Bài 1

I. Mục đích

- Giao tiếp với màn hình LCD với Atmega128.
- Điều khiển tốc độ động cơ DC.
- Hiển thị văn bản lên màn hình LCD 16x2.

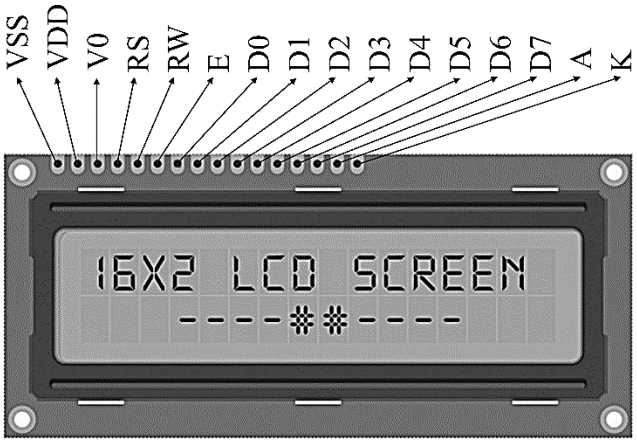
II. Phần cứng chủ yếu sử dụng trong bài thực hành

Trong bài thực hành này, chúng ta sử dụng các nút bấm Button, hiển thị tín hiệu với màn hình LCD, rơle, động cơ.

III. Tóm tắt nội dung lý thuyết

* ***Màn hình LCD 16x2***

Màn hình LCD 16x2 là một thiết bị ngoại vi dùng để giao tiếp với người dùng một cách trực quan dựa trên công nghệ tinh thể lỏng. Các tinh thể lỏng trong màn hình được kích thích bởi các tín hiệu điện, từ đó thay đổi cách chúng phản xạ ánh sáng để tạo ra các ký tự và hình ảnh trên màn hình. Màn hình có khả năng hiển thị 16 ký tự trên mỗi dòng và tổng cộng 2 dòng, có tích hợp đèn nền giúp đọc thông tin dễ dàng hơn trong điều kiện ánh sáng yếu. Khi sản xuất LCD 16x2, nhà sản xuất đã tích hợp chíp điều khiển loại 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 Hình 1.2.



Hình 1.2. Giao diện của màn hình LCD 16x2 thông thường

Chức năng các chân của màn hình LCD 16x2 được mô tả như ở Bảng 1.1 dưới đây.

Bảng 1.1. Chức năng của các chân màn hình LCD 16x2

Chân	Ký hiệu	Mô tả
1	VSS	Chân nối đất cho LCD 16x2, khi thiết kế mạch ta nối chân này với GND của bảng mạch.
2	VDD	Chân cấp nguồn cho LCD 16x2, khi thiết kế mạch ta nối chân này với nguồn +5V của bảng mạch.
3	V0	Điện áp điều chỉnh độ tương phản của LCD 16x2.
4	RS	Chân chọn thanh ghi. Nối chân RS với logic “0” (GND) hoặc logic “1” (VCC) để chọn thanh ghi. – Logic “0”: Dữ liệu từ 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”: Dữ liệu từ DB0-DB7 sẽ nối với thanh ghi dữ liệu (DR) bên trong LCD 16x2.
5	RW	Chân chọn chế độ đọc/ghi (Read/Write). Nối chân RW với logic “0” để LCD 16x2 hoạt động ở chế độ ghi, hoặc nối với logic “1” để LCD 16x2 ở chế độ đọc.
6	E	Chân cho phép (Enable). Sau khi các tín hiệu được đặt lên dữ liệu từ DB0-DB7, các lệnh chỉ được chấp nhận khi có 1 xung cho phép của chân E. – Ở chế độ ghi: Dữ liệu ở sẽ được LCD 16x2 chuyển vào thanh ghi bên trong nó khi phát hiện một xung ở sườn âm của tín hiệu chân E. – Ở chế độ đọc: Dữ liệu sẽ được LCD 16x2 xuất ra DB0-DB7 khi phát hiện một xung ở sườn dương của tín hiệu chân E và được LCD 16x2 giữ đến khi nào chân E xuống mức thấp.
7 đến 14	D0 đến D7	Tám đường dữ liệu dùng để trao đổi thông tin với máy tính. Có 2 chế độ sử dụng 8 đường dữ liệu này:

		– Chế độ 8 bit: Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit D7. – Chế độ 4 bit: Dữ liệu được truyền trên 4 đường từ D4 tới D7, bit MSB là D7
15	A	Nguồn dương cho đèn nền LCD 16x2.
16	K	GND cho đèn nền LCD 16x2.

***Nút bấm**

Nút bấm là phần tử lối vào kỹ thuật số đơn giản nhất, cho phép người dùng tương tác với vi điều khiển bằng thao tác nhấn/thả. Khi nhấn nút, tiếp điểm đóng lại tạo đường dẫn dòng điện, khi thả nút, tiếp điểm mở ra và mạch bị ngắt. Trên bảng mạch ATmega128, nút bấm được nối tới các chân I/O, do đó trạng thái của chúng được đọc thông qua các thanh ghi PINx. Về mặt logic, có hai kiểu mắc cơ bản là nối nút với mức logic thấp (GND) hoặc nối với mức logic cao (VCC).

***Rơ-le điện từ**

Rơ-le điện từ được dùng để điều khiển các tải tiêu thụ có công suất lớn hơn nhiều so với khả năng cấp dòng trực tiếp từng chân I/O của vi điều khiển. Về cấu tạo, rơ-le gồm một nam châm điện và hệ thống tiếp điểm cơ khí. Khi có dòng chạy qua cuộn dây, lực từ sinh ra sẽ hút cần gạt, làm đóng hoặc mở các tiếp điểm, nhờ đó ta có thể dùng tín hiệu điều khiển điện áp thấp để đóng cắt mạch điện cao hoặc dòng lớn hơn một cách gián tiếp, đồng thời tạo lớp cách ly an toàn giữa khối điều khiển và tải.

***LED**

LED là đi-ốt phát quang, viết tắt của "Light Emitting Diode" phát sáng khi được phân cực thuận. Điện áp rơi thuận của LED thường trong khoảng 1,5-3,3 V, dòng làm việc điển hình 10-30 mA tùy loại và màu. Trên bảng mạch thực hành, LED được dùng chủ yếu với vai trò đèn báo trạng thái, đèn nền của các màn hình hiển thị để hỗ trợ người dùng đọc được thông tin trong điều kiện thiếu ánh sáng.

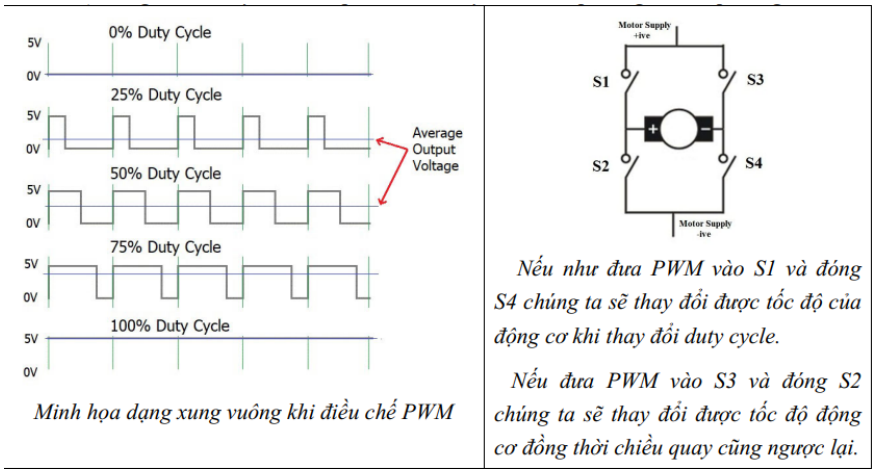
***Timer**

Timer/Counter là một bộ đếm xung nhịp, có thể là xung nhịp nội (timer) hoặc lấy từ ngoại vi (counter). Nó chứa các thanh ghi chứa giá trị đếm và thanh ghi điều khiển hoạt động của bộ đếm đó. Người dùng có thể lập trình cho Timer/Counter bắt đầu đếm từ 1 giá trị nào đó bằng cách ghi giá trị đó vào thanh ghi chứa giá trị đếm. Khi giá trị

đếm vượt quá giá trị tối đa mà thanh ghi chứa giá trị của Timer/Counter thì xuất hiện hiện tượng gọi là "tràn (overflow)"

Động cơ DC (điều chế PWM)

Hiểu một cách đơn giản thì điều chế PWM là thay đổi duty cycle (khoảng thời gian tín hiệu ở mức cao) trong một chu kỳ cố định, qua đó làm thay đổi điện áp trung bình cấp ra ngoại vi.

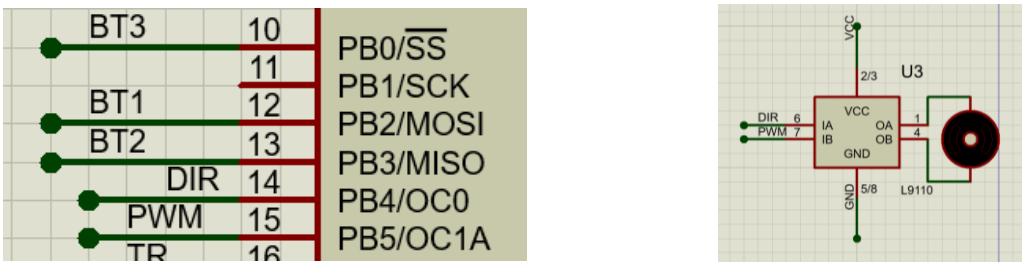


Hình 1.3. Điều chế PWM

IV. Nội dung thực hành

Thiết kế một chiếc máy giặt với các chức năng. Hiển thị các phím chức năng trên LCD (1-giặt; 2-xả; 3-vắt). Bấm BT1 – thực hiện chế độ giặt, ban đầu rơle 1 bật trong 2s (biểu thị quá trình cung cấp nước từ van nước), sau đó động cơ sẽ quay nhanh dần (sau mỗi giây, tốc độ động cơ tăng lên 20% - tương đối biểu thị quá trình giặt), sau 10s động cơ dừng quay, rơle 2 bật trong 2s (biểu thị quá trình xả nước thải), sau đó động cơ sẽ quay với tốc độ 100% trong vòng 3s để vắt.

1. Sơ đồ kết nối

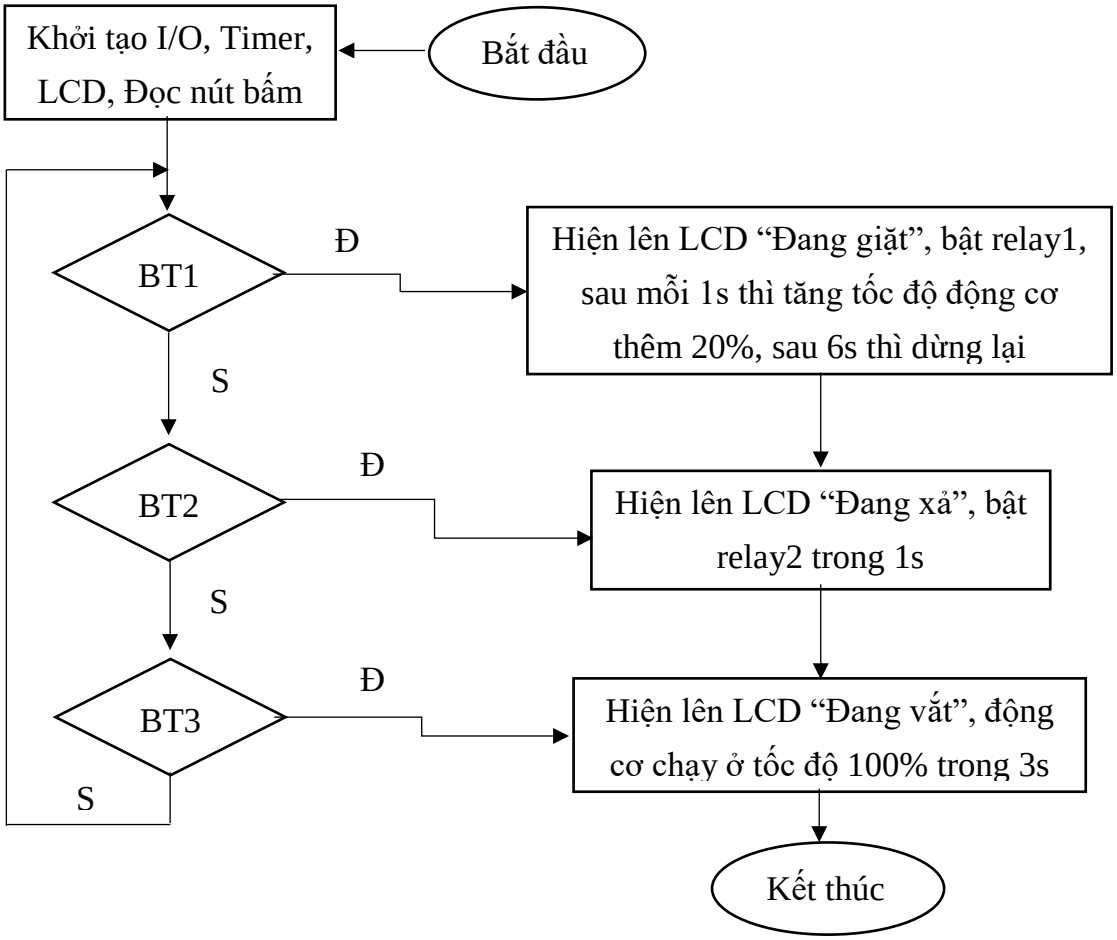


Hình 1.4. Sơ đồ kết nối chân motor

Ba nút bấm được nối với các chân PB0, PB2, PB3, và đầu còn lại được nối đất. Như vậy khi bấm nút thì điện áp nhận được là mức 0. Trong bài này ta dùng Timer1 tạo

xung PWM, do đó chân PWM nối với lối ra của Timer1, là chân OC1A. Chân DIR dùng để điều khiển chiều quay của motor.

2. Sơ đồ thuật toán



3. Mã chương trình

```
#include <io.h>
#include <alcd.h>
#include <stdio.h>
#include <delay.h>
#define BT1 PINB.2
#define BT2 PINB.3
#define BT3 PINB.0
#define LED PORTD.5
int dem=0,speed=20,x=0;
void dkmotor(int chieu, int toc_do)
{
   if(chieu==1)
   {
      PORTB.4=1;
OCR1A=255-255*toc_do/100;
   }
   if(chieu==0)
   {
if(x==3)
  {
     PORTC.2=0;
     lcd_clear();
     lcd_gotoxy(0,0);
     lcd_putsf("Dang vat");
     dkmotor(1,100);
     if(dem==1500)
     {
        dem=0;
        x=4;
     }
  }
  if(x==4)
   {
     lcd_clear();
     lcd_gotoxy(0,0);
     lcd_putsf("Xong");
```

```
        PORTB.4=0;
        OCR1A=255*toc_do/100;
    }
}
interrupt [TIM0_OVF] void timer0_ngat(void)
{
    TCNT0=0x06;
    dem=dem+1;
    if(x==1)

    {

        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_putsf("Dang giat");
        PORTC.3=1;
        if (dem==500)
        {
              speed=speed+20;
              dkmotor(0,speed);
        }
    if(dem%500==0 && speed<=100)
        {
              speed=speed+20;
              dkmotor(0,speed);
        }
        if(dem==3000)
        {
           dem=0;
           x=2;
        }
    }
    if(x==2)
        {
             lcd_clear();
             lcd_gotoxy(0,0);
             lcd_putsf("Dang xa");
             dkmotor(0,0);
             PORTC.2=1;
             if(dem==500)
             {
                dem=0;
                x=3;
             }
        }
        dkmotor(0,0);
        x=5;
        dem=0;
     }
}
void main(void)

{
   DDRB=0b00110000;
   PORTB=0b00101101;
TCCR1A=(1<<WGM10|1<<COM1A1);
TCCR1B=(1<<WGM12|1<<CS11|1<<CS10);
   TCCR0=0x03;
   TCNT0=0x06;
   OCR0=0x00;
   TIMSK=0x01;
   #asm("sei")
   DDRD.7=1;
   PORTD.7=1;
   DDRC=1<<3|1<<2;
   lcd_init(16);
   lcd_gotoxy(0,0);

   lcd_putsf("1-giat 2-xa 3-vat");
while (1)
   {
      if(BT1==0)
      {
         delay_ms(250);
         x=1;
         dem=0;
      }
      if(BT2==0)
      {
         delay_ms(250);
         x=2;
         dem=0;
      }
      if(BT3==0)
      {
         delay_ms(250);
         x=3;
         dem=0;
      }
   }
}
```

4. Hướng dẫn thực hành

Timer0 được cấu hình làm bộ đếm thời gian như sau:

- Dòng 92: chia tần số 64, và lấy tần số 8 MHz (không khai báo bit AS0 của thanh ghi ASSR0 nhưng ta hiểu là lấy tần số nội). Như vậy một xung có độ dài 64 ÷ 8MHz = 8us.

- Dòng 93: Thanh ghi TCNT0 được đặt ở giá trị 6. Nghĩa là từ giá trị 6 cho đến lúc tràn (256) là 250 lần. Như khoảng cách giữa 2 lần tràn liên tiếp là 2ms.

- Dòng 95: Bit TOIE0 được set lên 1 cho phép ngắt tràn Timer0.

- Dòng 96: Bit "Global Interrupt Enable" của thanh ghi SREG được set lên 1 cho phép ngắt toàn cục.

- Hàm phục vụ ngắt Timer0 (từ dòng 24): Mỗi khi hàm ngắt được gọi thì nó sẽ đếm 1 lần và đặt thanh ghi TCNT0 về giá trị 6. Như vậy 500 lần đếm thì sẽ được 1s. Khi đếm đủ thời gian thì thực thi chương trình trong cấu trúc *for* tuỳ theo yêu cầu của bài.

Timer1 được cấu hình tạo sóng PWM ở lối ra tương ứng:

- Dòng 90, 91: Bit WGM10 và WGM12 được set lên 1 cho phép Timer1 hoạt động ở chế độ Fast PWM 8-bit. Ở chế độ này thanh ghi OCR1A có giá trị trong khoảng từ 0 đến 255, và TCNT1 cũng đếm trong dải giá trị tương ứng.

- Bit COM1A1 và COM1A0 lần lượt là 1 và 0, theo như mô tả trong datasheet thì khi TCNT1L compare match với OCR1A, lối ra OC1A được đặt về 0, còn khi TCNT1 đạt TOP (255), lối ra OC1A được set lên 1. Nói 1 cách dễ hiểu thì khi TCNT1 > OCR1A, lối ra đặt về 0, khi TCNT1 < OCR1A, lối ra set lên 1. Như vậy giá trị của thanh ghi OCR1A sẽ quyết định duty circle của Timer1.

- 2 bit CS11 và CS10 dùng để thiết lập chu kì xung của lối ra, phần này ta không cần đi chi tiết vì đối với chế độ PWM thì duty circle có ý nghĩa hơn.

Sau khi nạp chương trình, màn hình sẽ hiện lên dòng chữ: "1-giat 2-xa 3-vat". Có nghĩa là bấm BT1 thì thực hiện chức năng giặt, bấm BT2 thì thực hiện chức năng xả, bấm BT3 thì thực hiện chức năng vắt. Khi bấm nút nào thì biến x sẽ nhận giá trị tương ứng để thực thi chế độ tương ứng, đồng thời biến *dem* sẽ đặt về 0.

Chế độ tương ứng với nút bấm được đặt trong hàm ngắt tràn của Timer0 (dòng 24-85):

- X=1: Chế độ giặt hiện lên màn hình LCD. Sau mỗi 1 giây (khi biến *dem* bằng 1 số nguyên lần 500), động cơ tăng tốc thêm 20%, tối đa 100%. Sau 6 giây (*dem=3000*), đặt lại *dem=0* và x=2.

- X=2: Chế độ xả hiện lên màn hình. Động cơ dừng, relay1 bật. Sau 1 giây đặt lại *dem=0* và x=3.

- X=3: Chế độ vắt hiện lên màn hình. Relay1 tắt, động cơ quay tốc độ 100%. Sau 3 giây đặt lại *dem=0* và *x=4*.

- X=4: Động cơ dừng, hiện lên màn hình đã xong một chu trình giặt.

Bài 2

I. Mục đích

- Giao tiếp với bàn phím ma trận, màn hình LCD với Atmega128.
- Điều khiển tốc độ động cơ servo.
- Hiển thị văn bản lên màn hình LCD 16x2.

II. Phần cứng chủ yếu sử dụng trong bài thực hành

Trong bài thực hành này, chúng ta sử dụng các nút bấm Button, bàn phím ma trận 3x3 hiển thị tín hiệu với màn hình LCD, động cơ servo.

III. Tóm tắt nội dung lý thuyết

* ***Màn hình LCD 16x2***

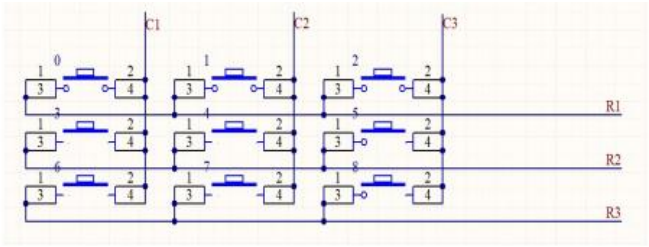
Đã trình bày ở bài 1

****Nút bấm***

Nút bấm là phần tử lối vào kỹ thuật số đơn giản nhất, cho phép người dùng tương tác với vi điều khiển bằng thao tác nhấn/thả. Khi nhấn nút, tiếp điểm đóng lại tạo đường dẫn dòng điện, khi thả nút, tiếp điểm mở ra và mạch bị ngắt. Trên bảng mạch ATmega128, nút bấm được nối tới các chân I/O, do đó trạng thái của chúng được đọc thông qua các thanh ghi PINx. Về mặt logic, có hai kiểu mắc cơ bản là nối nút với mức logic thấp (GND) hoặc nối với mức logic cao (VCC).

****Keypad 3x3***

Bàn phím ma trận được tạo thành từ các nút được sắp xếp theo hàng và cột. Mỗi nút nằm chính xác tại vị trí giao nhau của hàng và cột.



Hình 2.1. Cách các dây bên trong bàn phím ma trận 3×3 kết nối từng hàng với từng cột.

R là viết tắt của Row (Hàng); R1, R2, R3 là các chân đại diện cho Hàng 1, Hàng 2, Hàng 3 của bàn phím.

C là viết tắt của Column (Cột); C1, C2, C3 là các chân đại diện cho Cột 1, Cột 2, Cột 3 của bàn phím.

Mỗi nút nhấn trên bàn phím ma trận được đặt tại giao điểm của một Hàng và một Cột.

Khi một nút nhấn được bấm, nó sẽ tạo kết nối điện giữa chân Hàng tương ứng và chân Cột tương ứng.

Chân R1, R2, R3, C1, C2, C3 được kết nối với các chân GPIO (General Purpose Input/Output) của vi điều khiển để thực hiện quá trình quét phím (keypad scanning):

Quét Hàng (Outputs): Vi điều khiển lần lượt đặt trạng thái LOW (hoặc HIGH, tùy thuộc vào cách mắc) lên từng chân R (R1, R2, R3...) trong khi các chân Hàng khác giữ ở trạng thái HIGH (hoặc Z/input). Các chân R hoạt động như Đầu ra (Output).

Đọc Cột (Inputs): Vi điều khiển đọc trạng thái của các chân C (C1, C2, C3...). Các chân C thường được cấu hình với trở kéo lên (pull-up resistors) và hoạt động như Đầu vào (Input).

Xác định phím: Nếu một phím được bấm, tín hiệu LOW từ chân Hàng đang được kích hoạt sẽ truyền qua nút nhấn và làm cho chân Cột tương ứng chuyển sang trạng thái LOW (hoặc ngược lại).

Bằng cách biết Hàng nào đang được kích hoạt (R1, R2 hay R3...) và Cột nào nhận được tín hiệu thay đổi (C1, C2 hay C3...), vi điều khiển sẽ xác định được chính xác phím nào đã được nhấn.

Timer

Đã trình bày ở bài trước

Động cơ servo

Đã trình bày ở bài trước

IV. Nội dung bài thực hành

Xây dựng phần cứng, phần mềm cho hệ thống nhúng đơn giản mô phỏng chức năng chiếc thang máy:

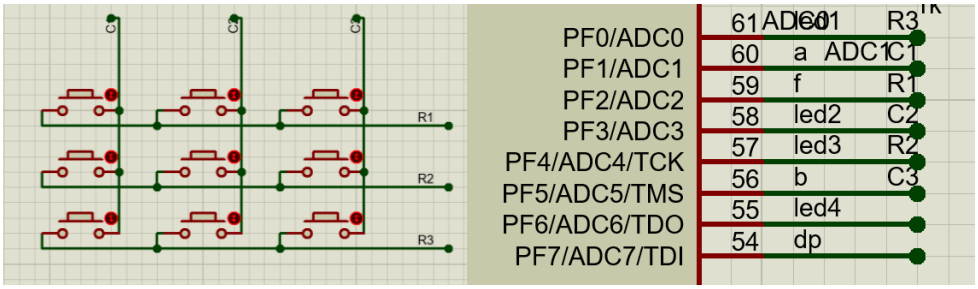
Thiết kế thang máy cho nhà 5 tầng thực hiện các chức năng sau (BT1 – lên, BT2 – xuống (ở ngoài thang), các phím 1-2-3-4-5 tương ứng với số tầng, 6 –mở, 7 đóng, 8 – khẩn cấp); Hiển thị số tầng lên LCD/GLCD/LED 7 đoạn.

Khi bấm cửa tầng nào thì động cơ servo sẽ xoay từ góc 0 - 90 độ (chú ý khi đang ở tầng bất kỳ không trùng với tầng tương ứng với phím bấm thì quay động cơ để về tầng tương ứng) nếu số tầng bấm nhỏ hơn số tầng đang phục vụ thì động cơ quay chiều kim đồng hồ, số đếm giảm. Nếu số tầng bấm lớn hơn số tầng đang phục vụ thì động cơ quay chiều ngược lại và số đếm tăng. Khi số đếm bằng số tầng được bấm, thì động cơ dừng lại

Sau khi động cơ servo quay sang góc 90 độ thì dừng lại trong vòng 2s rồi lại quay về góc 0 độ. nếu số tầng bấm cao hơn số tầng hiện tại thì động cơ quay ngược kim đồng hồ và số đếm tăng; nếu số tầng nhỏ hơn tầng hiện tại thì động cơ quay theo chiều kim đồng hồ và số đếm giảm. số đếm tăng hoặc giảm khi bằng số bấm thì động cơ dừng lại, servo quay về góc 90 độ trong 2s và lại quay về góc 0 độ. Quá trình này lặp đi lặp lại.

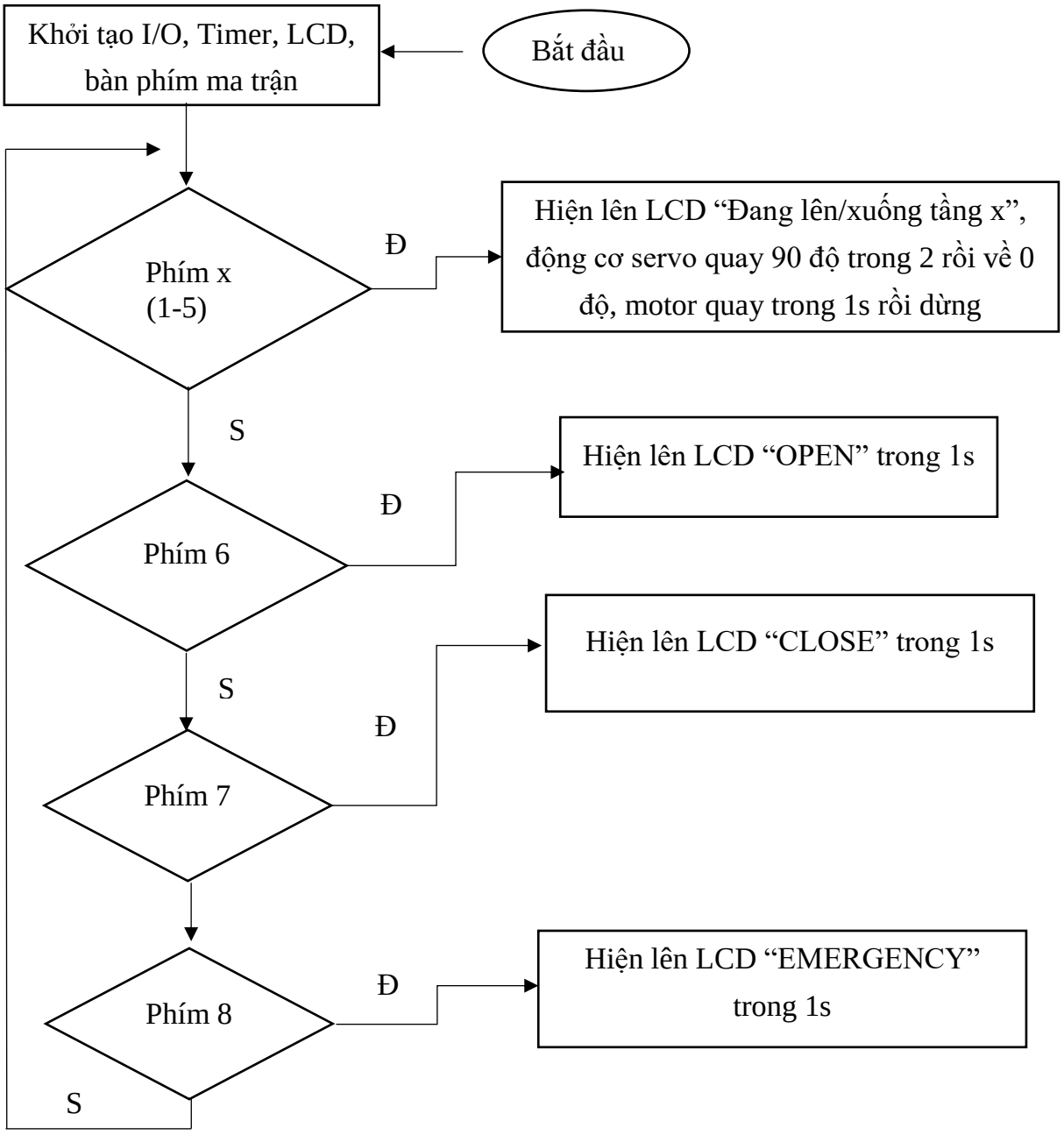
Trong quá trình thang máy đang hoạt động, LCD sẽ hiển thị quá trình này (ví dụ, đang ở tầng x và đang đi lên)

1. Sơ đồ kết nối



Hình 2.2. Sơ đồ kết nối bàn phím ma trận

2. Sơ đồ thuật toán



3. Mã chương trình

```
1   #include <io.h>
2   #include <delay.h>
3   #include <alcd.h>
4   #include <stdio.h>
5   #define servo_1 PORTC.7
6   int
    tangx=1,tangy=1,dem=0,vi_tri=
    10
7   char str[20];
8   void dkmotor(int chieu, int
    toc_do)
79  {
80      lcd_clear();
81      lcd_gotoxy(0,0);
82      lcd_putsf("CLOSE");
83      delay_ms(1000);
84  }

85  else if(PINF.3==0)
86  {
```

```
{
    if(chieu==1)
    {
        PORTB.4=1;
        OCR1A=255-255*toc_do/100;
    }
    if(chieu==0)
    {
        PORTB.4=0;
        OCR1A=255*toc_do/100;
    }
}
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
    TCNT0=0xB0;
    dem=dem+1;
    if (dem==2000) dem=0;
    if (dem<vi_tri) servo_1=1;
    else servo_1=0;
}
void button ()
{
    int i;
    for(i=0;i<3;i++)
        {
            if(i==0) PORTF=0b00111011;
            if(i==1) PORTF=0b11101111;
            if(i==2) PORTF=0b11111110;
            if(PORTF==0b00111011)
            {
                if (PINF.1==0)
                {
                    tangy=1;
                    delay_ms(100);
                }
                if(PINF.3==0)
                    lcd_clear();
                    lcd_gotoxy(0,0);
                    lcd_putsf("EMERGENCY");
                    delay_ms(1000);
                }
            }
        }
}
void main(void)
{
    TCCR1A=(1<<WGM10|1<<COM1A1);
    TCCR1B=(1<<WGM12|1<<CS11|1<<CS10);
    DDRB=(1<<4|1<<5);
    ASSR=0x00;
    TCCR0=0x01;
    TCNT0=0xB0;
    TIMSK=0x01;
    #asm("sei")
    DDRF=0b11010101;
    DDRD.7=1;
    PORTD.7=1;
    DDRC.3=1;
    lcd_init(16);
while (1)
    {
        button();
        if(tangx<tangy)
        {
            lcd_clear();
            sprintf(str,"level:%d to:%d",tangx,tangy);
            lcd_gotoxy(0,0);
            lcd_puts(str);
            lcd_gotoxy(0,1);
            lcd_putsf("up");
            vi_tri=150;
            delay_ms(2000);
            vi_tri=10;
```

```
46              {
47                 tangy=2;
48                 delay_ms(100);
49              }
50              if(PINF.5==0)
51              {
52                tangy=3;
53                delay_ms(100);
54              }
55           }
56
    if(PORTF==0b11101111)
57           {
58              if (PINF.1==0)
59              {
60                 tangy=4;
61                 delay_ms(100);
62              }
63              else if(PINF.3==0)
64              {
65                 tangy=5;
66                 delay_ms(100);
67              }
68              else if(PINF.5==0)
69              {
70                 lcd_clear();
71                 lcd_gotoxy(0,0);
72                 lcd_putsf("OPEN");
73                 delay_ms(1000);
74              }
75           }
76
    if(PORTF==0b11111110)
77           {
78              if (PINF.1==0)
124             dkmotor(1,100);
125             tangx++;
126             delay_ms(1000);
127          }
128          else if(tangx>tangy)
129          {
130             lcd_clear();
131             sprintf(str,"level:%d
        to:%d",tangx,tangy);
132             lcd_gotoxy(0,0);
133             lcd_puts(str);
134             lcd_gotoxy(0,1);
135             lcd_putsf("down");
136             vi_tri=150;
137             delay_ms(2000);
138             vi_tri=10;
139             dkmotor(0,90);
140             tangx--;
141             delay_ms(1000);
142          }
143          else
144          {
145          dkmotor(1,0);
146          lcd_clear();
147          sprintf(str,"level: %d",tangx);
148          lcd_gotoxy(0,0);
149          lcd_puts(str);
150          vi_tri=150;
151          delay_ms(2000);
152          vi_tri=10;
153          delay_ms(1000);
154          }
155       }
156    }
```

4. Hướng dẫn thực hành

Timer1 sẽ điều khiển tốc độ quay của động cơ (đã trình bày ở bài 1).

Timer0 điều khiển động cơ servo:

- Tạo hàm ngắt tràn 0.01 ms và đếm 2000 lần. Biến *vi_tri* có giá trị từ 100 đến 200 chỉ góc quay của servo tương ứng với 0 đến 90 độ. Nếu biến *dem* < *vi_tri* thì lối ra servo (PORTC.7) bằng 1, ngược lại thì bằng 0. Khi đếm đủ 2000 lần thì đếm lại từ đầu. Như vậy ta đã tạo được một xung PWM có chu kì 20 ms mà độ rộng xung phụ thuộc vào biến

vi_tri. Nếu *vi_tri = 100* độ rộng xung 1 ms, góc quay 0 độ, *vi_tri = 150* thì độ rộng xung 1,5 ms, góc quay 90, *vi_tri=200* thì độ rộng xung 2 ms, góc quay 180 độ.

Sau khi nạp chương trình, màn hình sẽ hiện lên "level: 1". Hàm *button()* sẽ chạy liên tục để đọc giá trị nút bấm. Bấm một nút bất kì từ 1 đến 5 thì màn hình sẽ hiện lên "level 1 to x - up", với x là tầng tương ứng nút bấm. Nếu đang ở tầng cao mà bấm xuống tầng thấp hơn thì màn hình sẽ hiện lên "down" ở dòng thứ 2. Khi đang trong quá trình di chuyển thì động cơ quay, tầng cũng sẽ tăng/giảm tương ứng.

Bài 3

I. Mục đích

- Giao tiếp modul thời gian thực DS1307 với Atmega128.
- Điều khiển tốc độ động cơ DC, servo.
- Hiển thị văn bản lên màn hình LCD 16x2.

II. Phần cứng chủ yếu sử dụng trong bài thực hành

Trong bài thực hành này, chúng ta sử dụng các nút bấm Button, modul đồng hồ thời gian thực DS1307 hiển thị tín hiệu với màn hình LCD, động cơ DC, động cơ servo.

III. Tóm tắt nội dung lý thuyết

DS1307 là một chip Đồng hồ thời gian thực (RTC - Real Time Clock) giá rẻ, phổ biến, có khả năng theo dõi chính xác thời gian (giây, phút, giờ, ngày, tháng, năm) và giao tiếp với vi điều khiển qua giao thức I2C. Các tính năng chính:

- Đếm thời gian: Theo dõi giờ, phút, giây, thứ, ngày, tháng, năm, hỗ trợ cả định dạng 12 giờ (có AM/PM) và 24 giờ.

- Giao tiếp I2C: Sử dụng 2 chân (SDA, SCL) để đọc/ghi dữ liệu, giúp tiết kiệm chân tín hiệu cho vi điều khiển.

- Nguồn dự phòng: Tự động chuyển sang dùng pin (thường là pin CMOS) khi mất điện chính, giúp duy trì thời gian liên tục.

- Bộ nhớ RAM 56 byte: Tích hợp thêm bộ nhớ tạm để lưu trữ dữ liệu không bị mất khi tắt nguồn.

- Thạch anh nội: Tích hợp sẵn bộ dao động thạch anh 32.768kHz, làm cho mạch nhỏ gọn và hoạt động chính xác (với sai số thấp)

Hình 3.1. Module thời gian thực DS1307

IV. Nội dung bài thực hành

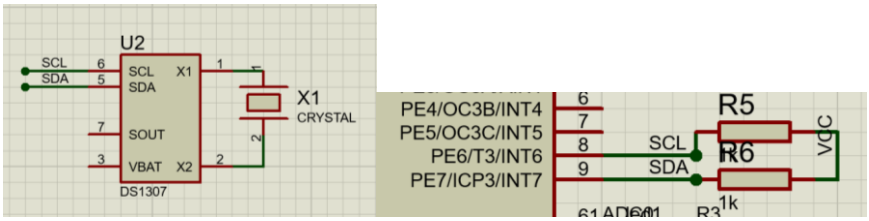
Thiết kế thiết bị cho cá ăn tự động với các chức năng sau (BT1, BT2, BT3 là các phím chức năng để hẹn giờ)

Thiết lập giờ, phút, chế độ trên LCD/glcd (tùy chọn một trong 2)

Khi đến giờ cho cá ăn, thì động cơ sẽ quay trong vòng 1s và servo sẽ mở góc từ 0 đến 90.

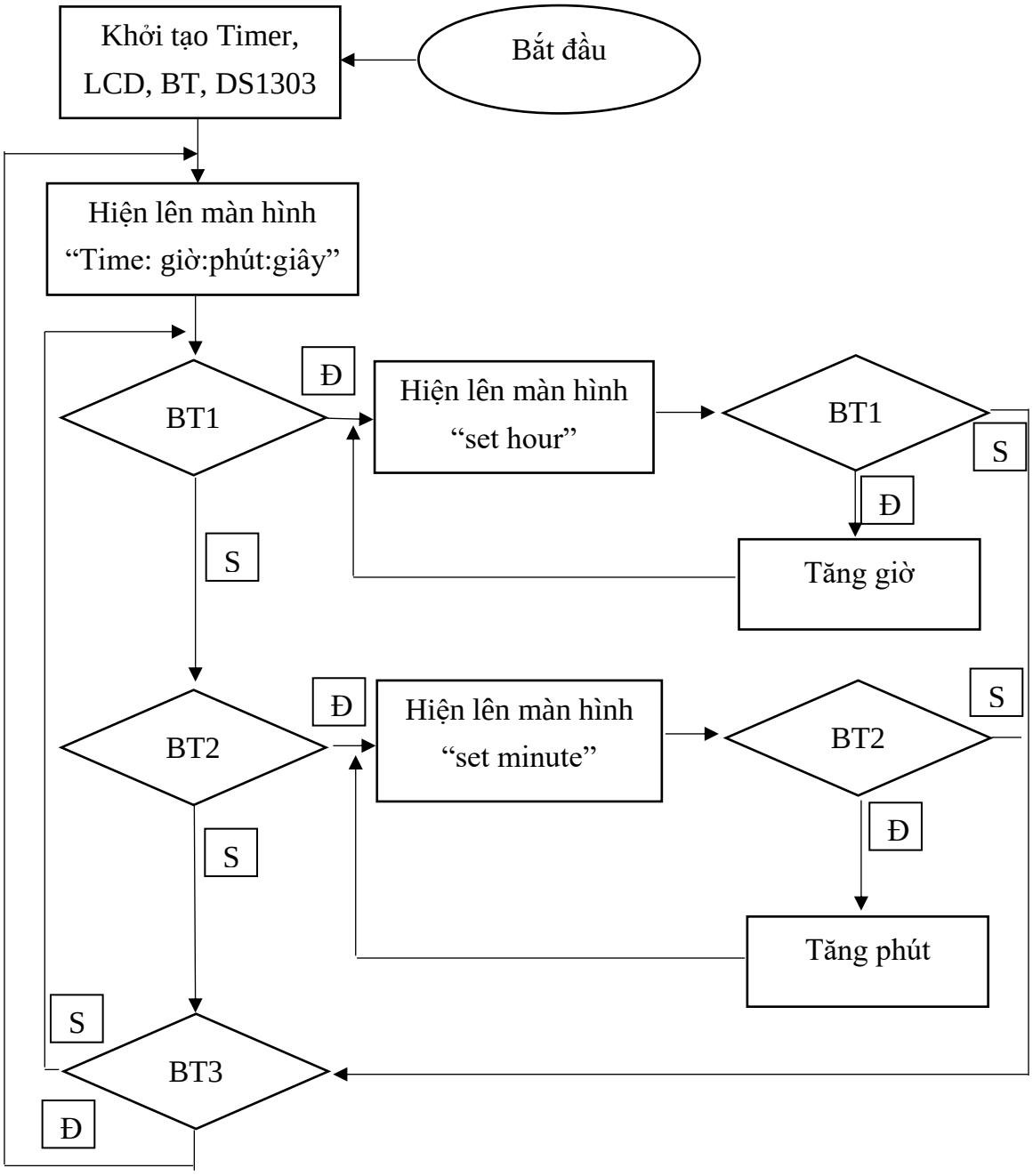
Bấm công tắc 1, đèn xanh đỏ sẽ nhấp nháy theo chu kỳ 1s và động cơ servo và động cơ dc sẽ ngừng ngay lập tức.

1. Sơ đồ kết nối



Hình 2.2. Sơ đồ kết nối 2 chân SCL và SDA của module

2. Sơ đồ thuật toán



3. Mã chương trình

```
1   #include <io.h>
2   #include <ds1307.h>
3   #include <alcd.h>
4   #include <stdio.h>
5   #include <delay.h>
6   #define servo_1 PORTC.7
7   #define BT1 PINB.2
8   #define BT2 PINB.3
9   #define BT3 PINB.0
10  char display_buffer[17];
51  rtc_init(3,1,0);
52  rtc_set_time(12,59,50);
53  while (1)
54  {
55  if(x==0)
56  {
57  rtc_get_time(&hour,&min,&sec);
58  sprintf(display_buffer,"Time:%02d:%0
    2d:%02d\n",hour,min,sec);
59  lcd_clear();
60  lcd_puts(display_buffer);
```

```
unsigned char hour,min,sec,gio=13,phut=00;
int speed=75, dir=1,dem=0,vi_tri=100,x=0;
void dkmotor(int chieu, int toc_do)
{
   if(chieu==1)
   {
      PORTB.4=1;
      OCR1A=255-255*toc_do/100;
   }
   if(chieu==0)
   {

      PORTB.4=0;
      OCR1A=255*toc_do/100;
   }
}
interrupt [TIM0_OVF] void timer0_ngat(void)
{
   TCNT0=0xB0;

   dem++;
   if (dem==2000) dem=0;
   if (dem<vi_tri) servo_1=1;
   else servo_1=0;
}

void main(void)
{
   DDRD.7=1;
   PORTD.7=1;
   DDRC.7=1;

TCCR1A=(1<<WGM10|1<<COM1A1);

TCCR1B=(1<<WGM12|1<<CS11|1<CS10);
   DDRB=(1<<4|1<<5);
   PORTB=0b00001111;
   TCCR0=0x01;
if(hour==gio&&min==phut&&sec==00)
{

lcd_gotoxy(0,1);

lcd_putsf("Dang cho ca an");
dkmotor(dir,speed);
vi_tri=200;
delay_ms(1000);
dkmotor(0,0);

}
}
if(BT1==0) x=1;
if(x==1)
{
if(BT1==0)
{
gio=gio+1;
if(gio==24) gio=0;

}
sprintf(display_buffer,"set hour:%02d",gio);
lcd_clear();
lcd_puts(display_buffer);
delay_ms(200);
}
if(BT2==0) x=2;
if(x==2)
{
if(BT2==0)
{
phut=phut+1;
if(phut==60) phut=0;
}


sprintf(display_buffer,"set minute:%02d",phut);

lcd_clear();
lcd_puts(display_buffer);
delay_ms(200);
```

```
46      TCNT0=0xB0;
47      TIMSK=1<<TOIE0;
48      #asm("sei")
49      lcd_init(16);
50      i2c_init();
96      }
97      if(BT3==0) x=0;
98      }
99      }
```

4. Hướng dẫn thực hành

Trong bài này ta sử dụng Timer1 tạo PWM điều khiển motor quay, Timer0 điều khiển động cơ servo (Phần này đã được trình bày ở các bài trước).

Chương trình bắt đầu bằng việc khởi tạo màn hình LCD, nút bấm và ic DS1307 (dòng 38 – 52). Sau đó vi điều khiển sẽ lấy dữ liệu từ DS1307 thông qua giao thức i2c và hiện thời gian lên màn hình (dòng 57 – 60). Khi ta nhấn nút thì biến *x* thay đổi (dòng 71 – 97). Nếu nhấn BT1 thì *x=1,* chuyển sang chế độ đặt giờ, tiếp tục nhấn BT1 thì giờ thay đổi. Nếu nhấn BT2 thì *x=3,* chuyển sang chế độ đặt phút, tiếp tục nhấn BT2 thì phút thay đổi. Nếu nhấn BT3 thì *x=3,* quay trở lại chế độ hiện thời gian lên màn hình. Nếu thời gian chạy đến thời điểm được thiết lập thì hiện lên màn hình "Đang cho cá ăn". Lúc này động cơ servo và motor quay (dòng 61 – 69).

Bài 4

I. Mục đích

- Giao tiếp với bàn phím ma trận 3x3, màn hình LCD với Atmega128.
- Hiển thị văn bản lên màn hình LCD 16x2.

II. Phần cứng chủ yếu sử dụng trong bài thực hành

Trong bài thực hành này, chúng ta sử dụng bàn phím ma trận 3x3 hiển thị tín hiệu với màn hình LCD, LED và rơ-le.

III. Tóm tắt nội dung lý thuyết

Bàn phím ma trận

* ***Màn hình LCD 16x2***

Rơ-le điện từ

LED

IV. Nội dung thực hành

Thiết kế bộ cửa mã khóa phím, mật khẩu đúng là 666, nhập 3 lần sai mật khẩu sẽ bật role 1 và đèn xanh đỏ nhấp nháy theo chu kỳ 1s. (hiển thị lên LCD/GLCD tùy chọn)

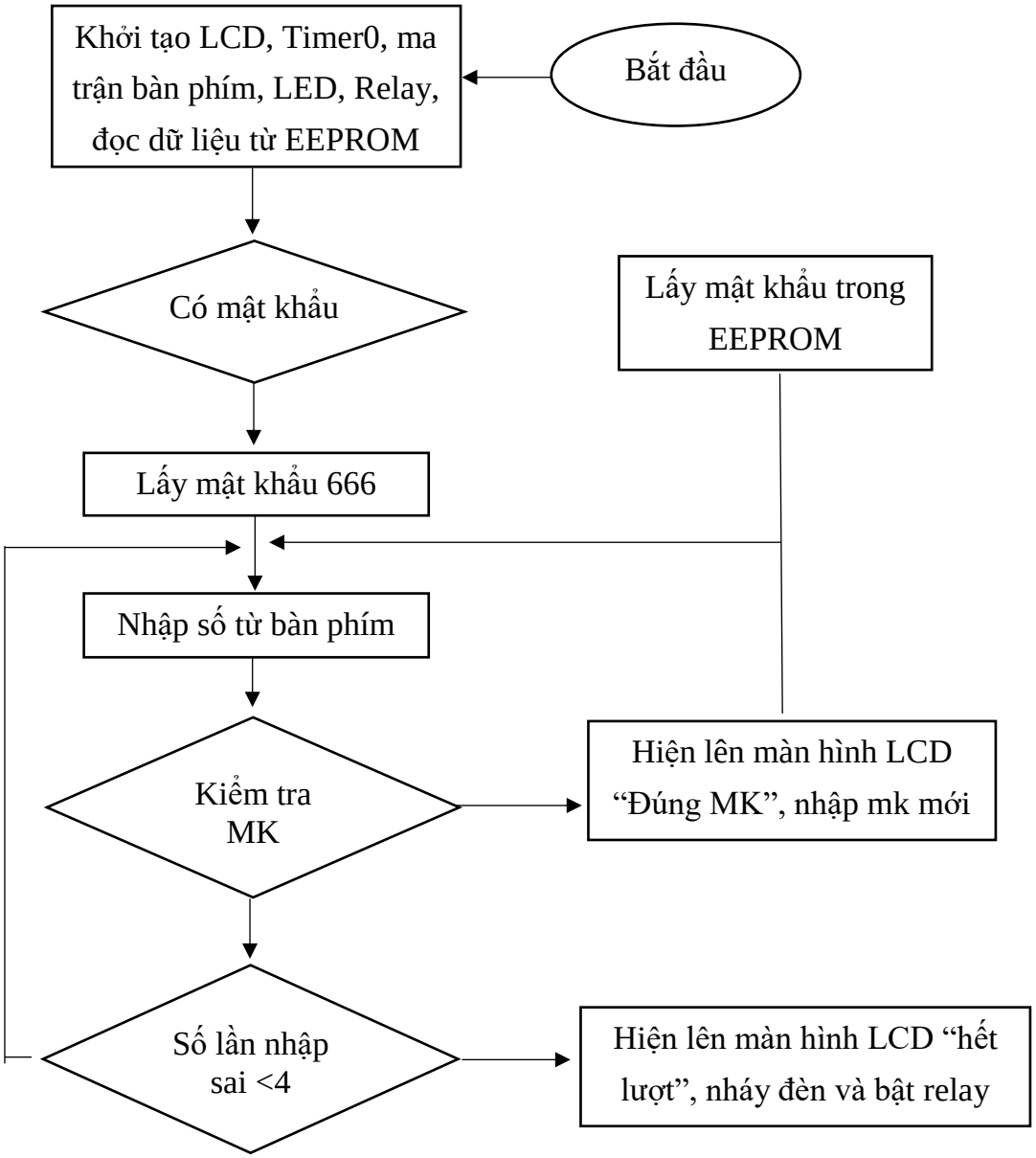
(tùy chọn: sau khi nhập đúng mật khẩu, cho phép người dùng đổi mật khẩu và lưu mật khẩu mới vào eeprom)

(giải thích thêm: nếu rút nguồn ra, cắm nguồn lại thì phải nhập mật khẩu mới đổi)

1. Sơ đồ kết nối

Kết nối bàn phím ma trận, LED, relay (đã trình bày ở các bài trước)

2. Sơ đồ thuật toán



3. Mã chương trình

```
#include <io.h>
#include <stdlib.h>
#include <stdio.h>
#include <delay.h>
#include <alcd.h>
#include <eeprom.h>
int mk[]={0,0,0},mk1,dem=0,dem1=0,nhapsai=2,kd=0,reset=0;
int EEMEM my_mk;
char str[15];
interrupt [TIM0_OVF] void timer0_ngat(void)
{
   TCNT0=0x06;
   if(nhapsai==0)
   if(PORTF==0b11111110)
   {
   if (PINF.1==0)
   {
   lcd_gotoxy(dem+3,0);
   lcd_putsf("6");
   delay_ms(500);

   lcd_gotoxy(dem+3,0);
   lcd_putsf("*");
   mk[dem]=6;

   dem++;
   delay_ms(1000);
   }
```

```
    {
      dem1=dem1+1;
      if (dem1==1000)
      {
        PORTD.5=1;
        PORTC.3=1;
      }
      if (dem1==2000)  PORTD.5=0;
      if (dem1==3000)  PORTD.4=1;
      if (dem1==4000)  PORTD.4=0;
      if (dem1==5000)  PORTD.6=1;
      if (dem1==6000)
      {
        PORTD.6=0;
        PORTC.3=0;
        dem1=0;
        nhapsai=4;
      }
    }
}
void mbutton()
{
  int i;
  for(i=0;i<3;i++)
    {
      if(i==0) PORTF=0b00111011;
      if(i==1) PORTF=0b00101111;
      if(i==2) PORTF=0b11111110;
      if(PORTF==0b00111011)
      {
        if (PINF.1==0)
        {
          lcd_gotoxy(dem+3,0);
          lcd_putsf("0");
          delay_ms(500);
          lcd_gotoxy(dem+3,0);
          lcd_putsf("*");
          mk[dem]=0;
          dem++;

          delay_ms(1000);
        }
       if(PINF.3==0)
       {
          lcd_gotoxy(dem+3,0);
          lcd_putsf("1");
          delay_ms(500);
          lcd_gotoxy(dem+3,0);
```

```
      else if(PINF.3==0)
            {
        lcd_gotoxy(dem+3,0);
          lcd_putsf("7");
          delay_ms(500);
        lcd_gotoxy(dem+3,0);
          lcd_putsf("*");
           mk[dem]=7;
             dem++;
          delay_ms(1000);
             }
      else if(PINF.5==0)
             {
        lcd_gotoxy(dem+3,0);
          lcd_putsf("8");
          delay_ms(500);
        lcd_gotoxy(dem+3,0);
          lcd_putsf("*");
           mk[dem]=8;
             dem++;
          delay_ms(1000);
             }
             }
             }
             }
        void main(void)
             {
        ASSR=0<<AS0;
          TCCR0=0x03;
          TCNT0=0X06;
            OCR0=0;
          TIMSK=0x01;
           #asm("sei")
       DDRF=0b11010101;
       DDRD=0b11110000;
            DDRC.3=1;
           PORTD.7=1;
           lcd_init(16);
mk1=eeprom_read_word(&
                my_mk);
     if(mk1==-1) mk1=666;
             while (1)
             {
           if(kd==0)
             {
         lcd_gotoxy(0,0);
        lcd_putsf("MK:");
         lcd_gotoxy(0,1);
```

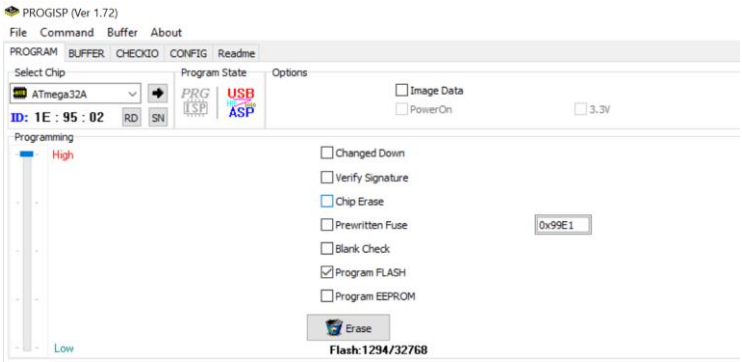
```
                lcd_putsf("*");
                mk[dem]=1;
                dem++;
                delay_ms(1000);
            }
            if(PINF.5==0)

            {
                lcd_gotoxy(dem+3,0);
                lcd_putsf("2");
                delay_ms(500);
                lcd_gotoxy(dem+3,0);
                lcd_putsf("*");
                mk[dem]=2;
                dem++;
                delay_ms(1000);
            }
        }
        if(PORTF==0b00101111)
        {
            if (PINF.1==0)
            {
                lcd_gotoxy(dem+3,0);
                lcd_putsf("3");
                delay_ms(500);
                lcd_gotoxy(dem+3,0);
                lcd_putsf("*");

                mk[dem]=3;
                dem++;
                delay_ms(1000);
            }
            else if(PINF.3==0)
            {
                lcd_gotoxy(dem+3,0);
                lcd_putsf("4");
                delay_ms(500);
                lcd_gotoxy(dem+3,0);
                lcd_putsf("*");
                mk[dem]=4;
                dem++;
                delay_ms(1000);
            }
            else if(PINF.5==0)
            {
sprintf(str,"mk cu:%d",mk1);
                lcd_puts(str);
                }
                mbutton();
            if (reset==0&&dem==3)
            {    if
((mk[0]*100+mk[1]*10+mk
[2])==mk1)
                {
                kd=1;
                reset=1;
                lcd_clear();
                lcd_gotoxy(0,0);
                lcd_putsf("Dung mk");
                dem=0;
                delay_ms(500);
                lcd_clear();
                lcd_gotoxy(0,0);
                lcd_putsf("new:");
                }
                else
                {
                nhapsai--;
                lcd_clear();
                lcd_gotoxy(0,0);
                lcd_putsf("Sai MK");
                lcd_gotoxy(0,1);
                sprintf(str,"con %d
lan",nhapsai);
                lcd_puts(str);
                delay_ms(1000);
                lcd_clear();
                }
                dem=0;
                if(nhapsai==0)
                {
                kd=1;
                lcd_clear();
                lcd_gotoxy(0,0);
                lcd_putsf("Het luot");
                }
                }
                if(reset==1)
                {
                mbutton();
                if(dem==3)
mk1=mk[0]*100+mk[1]*10
+mk[2];
```

```
104                 lcd_gotoxy(dem+3,0);
105                 lcd_putsf("5");
106                 delay_ms(500);
107                 lcd_gotoxy(dem+3,0);
108                 lcd_putsf("*");
109                 mk[dem]=5;
110                 dem++;
111                 delay_ms(1000);
112             }
113         }
```

```
217     eeprom_update_word(&my_
            mk,mk1);
218     if(dem==3)
219         {
220     lcd_clear();
221     lcd_gotoxy(0,0);
222     lcd_putsf("Da luu");
223         }
224         }
225         }
226         }
```

4. Hướng dẫn thực hành

Để không bị xoá dữ liệu sau khi ngắt nguồn, xoá tích ở ô *Chip erase* trên Progisp:



Sau khi nạp chương trình, vi điều khiển sẽ đọc dữ liệu từ eeprom (dòng 165 – 166). Nếu có mật khẩu đã được lưu thì lấy nó làm mật khẩu, nếu không thì lấy 666 làm mật khẩu, sau đó sẽ hiện mật khẩu này lên dòng thứ 2 của màn hình LCD (dòng 169 – 176).

Nhập mật khẩu: Hàm *mbutton* được gọi trong vòng *while* để đọc dữ liệu từ bàn phím (dòng 177). Sau khi nhập 3 số sẽ kiểm tra số đã nhập với mật khẩu đã lưu (dòng 178 – 223). Nếu đúng thì cho phép thay đổi mật khẩu, ta tiếp tục nhập 3 số mới, sau đó lưu vào eeprom. Nếu nhập sai sẽ báo và cho nhập lại. Nếu sai 3 lần thì thông báo “hết lượt” và nháy đèn, bật relay (trong hàm ngắt tràn của Timer0).

Bài 5

I. Mục đích

- Sử dụng ADC đọc dữ liệu.
- Điều khiển động cơ.

II. Phần cứng chủ yếu sử dụng trong bài thực hành

***Màn hình LCD**

***Motor**

III. Tóm tắt lý thuyết

Bộ ADC (Analog-to-Digital Converter - Bộ chuyển đổi Tương tự-Số) trên ATmega128 là bộ chuyển đổi 10-bit, có 8 kênh đầu vào, giúp biến đổi tín hiệu điện áp analog từ cảm biến (như nhiệt độ, ánh sáng) thành giá trị số mà vi điều khiển có thể xử lý, với tốc độ chuyển đổi cao và hỗ trợ nhiều chế độ hoạt động linh hoạt, được tích hợp sẵn, là tính năng quan trọng cho giao tiếp cảm biến trong các ứng dụng nhúng.

Các đặc điểm chính của ADC ATmega128:

- Độ phân giải (Resolution): 10-bit, cho phép biểu diễn tín hiệu analog thành 1024 giá trị số (từ 0 đến 1023).

- Số kênh (Channels): 8 kênh analog đầu vào (ADC0-ADC7), cho phép đọc dữ liệu từ 8 cảm biến khác nhau.

- Chức năng: Chuyển đổi tín hiệu điện áp tương tự thành giá trị số 10-bit.

- Tích hợp: Tích hợp sẵn trong vi điều khiển, không cần chip ADC rời.

- Ứng dụng: Lý tưởng cho việc giao tiếp với các cảm biến (nhiệt độ, ánh sáng, khoảng cách, v.v.).

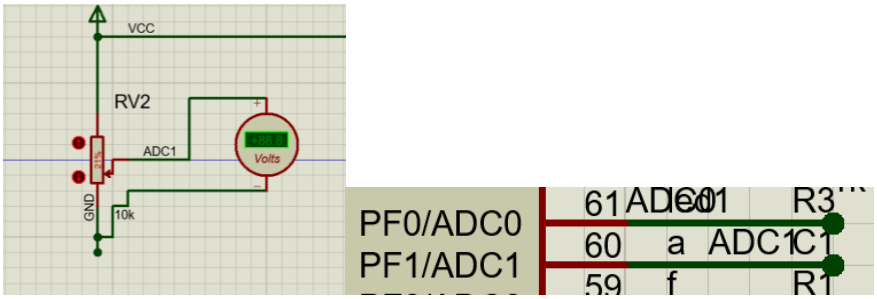
IV. Nội dung bài thực hành

Viết chương trình điều khiển xe điện thực hiện các chức năng sau, (hiển thị tốc độ tương đối lên LCD/GLCD)

Dùng chiết áp xoay theo chiều kim đồng hồ, ở vị trí giữa -> động cơ dừng. Xoay về phía max động cơ sẽ tăng dần tốc độ, nếu xoay về phía min động cơ sẽ tăng dần tốc độ nhưng ở chiều ngược lại.

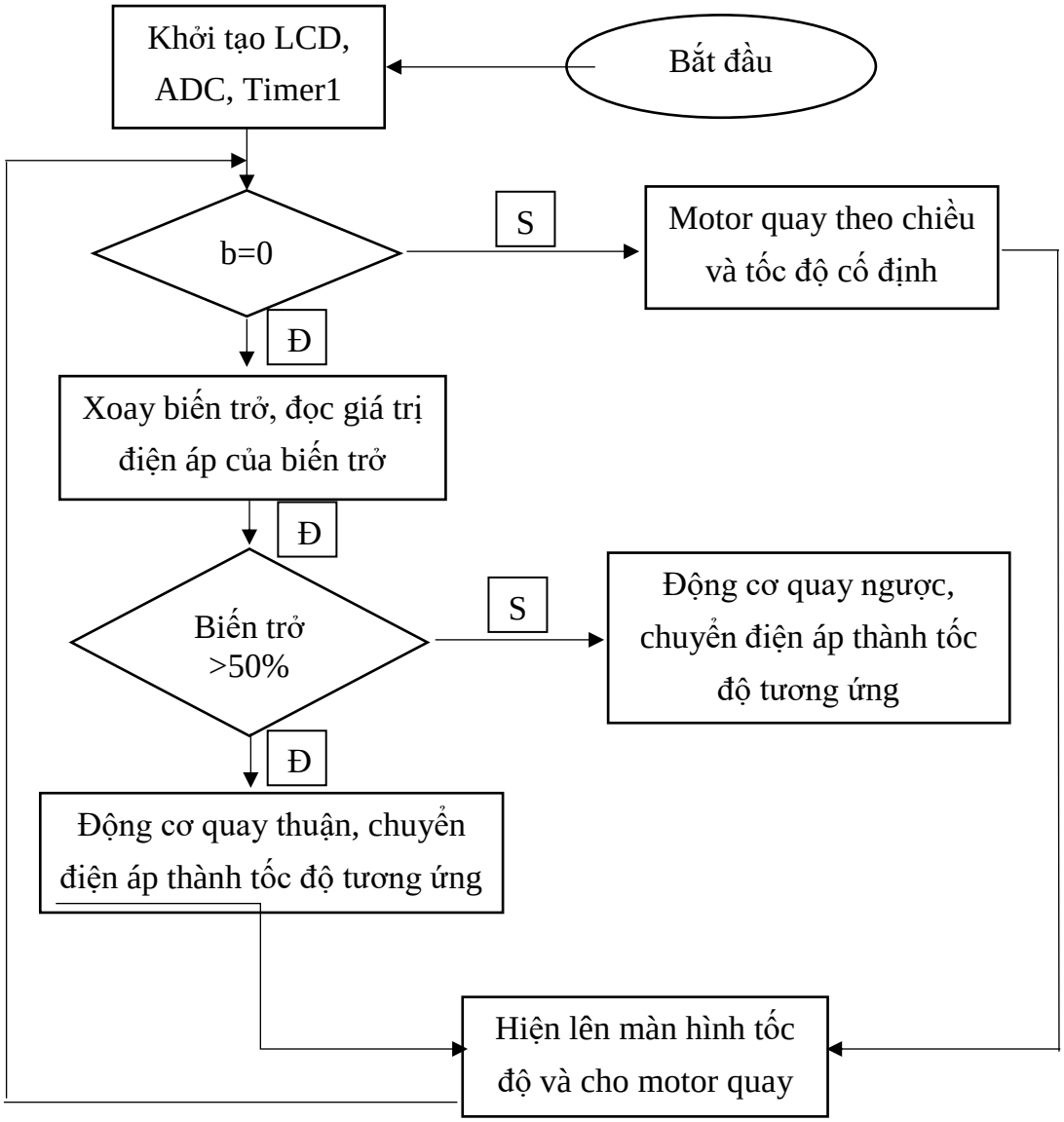
Nếu bấm BT1, động cơ sẽ quay ở 1 tốc độ cố định (không còn chịu ảnh hưởng của chiết áp). Nếu bấm BT1 một lần nữa, tốc độ của động cơ lại phụ thuộc vào chiết áp như lúc đầu.

1. Sơ đồ kết nối



Hình 5.1. Sơ đồ kết nối chân ADC1 vào biến trở

2. Sơ đồ thuật toán



3. Mã chương trình

```
#include <io.h>
#include <delay.h>
#include <alcd.h>
#include <stdio.h>
#define BT1 PINB.2

float speed=0;
unsigned int giatri;

char buffer[20];
bit b=0;
void dkmotor(int chieu, int toc_do)
{
if(chieu==1)
{
PORTB.4=1;
lcd_init(16);
DDRB=0b00110000;
PORTB=0b00101101;
TCCR1A=(1<<WGM10|1<<COM1A1);
TCCR1B=(1<<WGM12|1<<CS11|1<CS10);

ADMUX=0x01;
ADCSRA=(1<<ADEN|1<<ADPS1|1<<ADPS0);
while (1)
{
if(BT1==0)

{
b=(!b);
delay_ms(500);
}
```

```
15  OCR1A=255-255*toc_do/100;
16  }
17  if(chieu==0)
18  {
19
20  PORTB.4=0;
21  OCR1A=255*toc_do/100;
22  }
23  }
24  unsigned int read_adc()
25  {
26  ADMUX=0x01;
27  delay_us(10);
28  ADCSRA|=1<<ADSC;
29  while((ADCSRA & 0b00010000)==0);
30  ADCSRA|=0x10;
31  return ADCW;
32  }
33  void main(void)
34  {
35  DDRD.7=1;
36  PORTD.7=1;
51  if(b==0)
52  {
53  giatri=read_adc();
54  speed=(float) giatri*100/1023-50;
55  sprintf(buffer,"speed: %.0f %%",speed);
56  lcd_gotoxy(1,0);
57  lcd_puts(buffer);
58  lcd_clear();
59  if(speed>=0)
60  dkmotor(1,speed);
61  else
62  dkmotor(0,-speed);
63  }
64  if(b==1)
65  {
66  lcd_putsf("speed: 100%");
67  dkmotor(1,100);
68  lcd_clear();
69  }
70  }
71  }
72
```

4. Hướng dẫn thực hành

Thiết lập ADC đọc dữ liệu từ chân ADC1:

- Dòng 40: *ADMUX=0x01* nghĩa là ta đọc dữ liệu ở chân ADC1.

- Dòng 41: Bit ADEN được set lên 1 cho phép ADC hoạt động, 2 bit ADPS0 và ADPS1 được set lên 1 để chia tần số clock cho bộ ADC, tần số này là tần số lấy mẫu.

- Hàm *read_adc* (dòng 24 – 32): Ghi bit ADSC là 1 (dòng 28) để bắt đầu quá trình chuyển đổi. Khi 1 quá trình chuyển đổi hoàn tất thì bit này được ghi là 1. Như vậy ta đợi đến khi bit này được ghi là 0 thì ta sẽ đọc dữ liệu (dòng 29 - 30). Dữ liệu trả về dưới dạng 2 thanh ghi ADCL và ADCH, kết hợp 2 thanh ghi này (ADCW) ta thu được giá trị ADC dưới dạng 10 bit.

Ta quy ước biến *b: b=0* thì động cơ phụ thuộc biến trở, *b=1* thì động cơ không phụ thuộc biến trở. Sau khi nạp chương trình, vì biến *b* ban đầu bằng 0 nên động cơ phụ thuộc vào biến trở. Thay đổi giá trị biến trở ta sẽ thấy giá trị speed in trên màn hình LCD sẽ thay đổi tương ứng, và cả động cơ cũng thay đổi tốc độ (dòng 51 – 63).

Nhấn BT1 thì biến *b* đảo trạng thái bằng 1 (dòng 46 – 50), lúc này động cơ quay ở tốc độ và chiều cố định (dòng 64 – 69). Nhấn BT1 lần nữa thì *b* đảo trạng thái về 0, động cơ lại phụ thuộc biến trở.

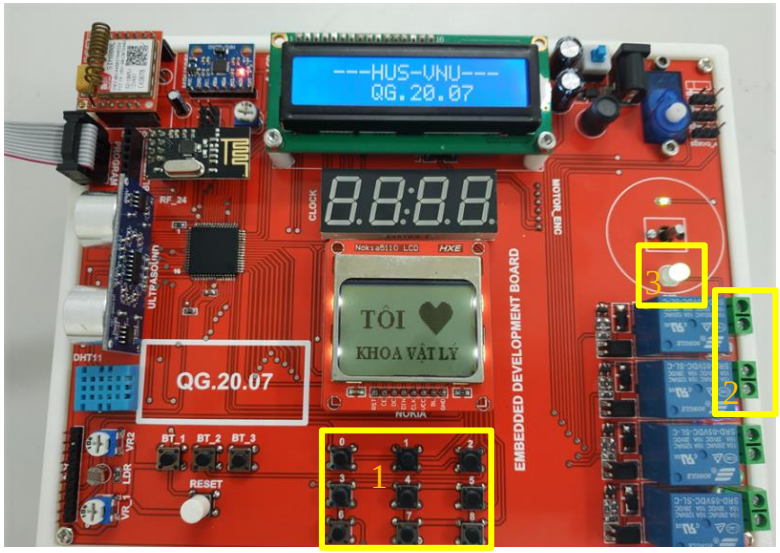
BÀI 6. GIAO TIẾP BÀN PHÍM MA TRẬN

I. Mục đích

– Giao tiếp với bàn phím ma trận với Atmega128.

II. Phần cứng chủ yếu sử dụng trong bài thực hành

Trong bài thực hành này, chúng ta sử dụng các nút bấm của bàn phím ma trận, hiển thị tín hiệu với các rơle 1, rơle 2 và đèn led RGB. Các phần cứng này có sẵn trên bảng mạch thực hành ở các vị trí 1, 2, 3 trên Hình 6.1.

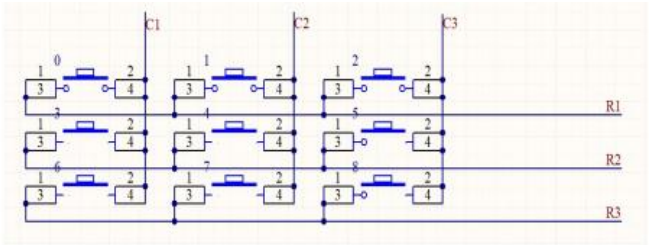


Hình 6.1. Phần cứng sử dụng trong bài thực hành 6

III. Tóm tắt nội dung lý thuyết

***Bàn phím ma trận**

Bàn phím ma trận được tạo thành từ các nút được sắp xếp theo hàng và cột. Mỗi nút nằm chính xác tại vị trí giao nhau của hàng và cột.



Hình 6.2. Cách các dây bên trong bàn phím ma trận 3×3 kết nối từng hàng với từng cột.

R là viết tắt của Row (Hàng); R1, R2, R3 là các chân đại diện cho Hàng 1, Hàng 2, Hàng 3 của bàn phím.

C là viết tắt của Column (Cột); C1, C2, C3 là các chân đại diện cho Cột 1, Cột 2, Cột 3 của bàn phím.

Mỗi nút nhấn trên bàn phím ma trận được đặt tại giao điểm của một Hàng và một Cột.

Khi một nút nhấn được bấm, nó sẽ tạo kết nối điện giữa chân Hàng tương ứng và chân Cột tương ứng.

Chân R1, R2, R3, C1, C2, C3 được kết nối với các chân GPIO (General Purpose Input/Output) của vi điều khiển để thực hiện quá trình quét phím (keypad scanning):

Quét Hàng (Outputs): Vi điều khiển lần lượt đặt trạng thái LOW (hoặc HIGH, tùy thuộc vào cách mắc) lên từng chân R (R1, R2, R3...) trong khi các chân Hàng khác giữ ở trạng thái HIGH (hoặc Z/input). Các chân R hoạt động như Đầu ra (Output).

Đọc Cột (Inputs): Vi điều khiển đọc trạng thái của các chân C (C1, C2, C3...). Các chân C thường được cấu hình với trở kéo lên (pull-up resistors) và hoạt động như Đầu vào (Input).

Xác định phím: Nếu một phím được bấm, tín hiệu LOW từ chân Hàng đang được kích hoạt sẽ truyền qua nút nhấn và làm cho chân Cột tương ứng chuyển sang trạng thái LOW (hoặc ngược lại).

Bằng cách biết Hàng nào đang được kích hoạt (R1, R2 hay R3...) và Cột nào nhận được tín hiệu thay đổi (C1, C2 hay C3...), vi điều khiển sẽ xác định được chính xác phím nào đã được nhấn.

Rơ-le điện từ

Rơ-le điện từ được dùng để điều khiển các tải tiêu thụ có công suất lớn hơn nhiều so với khả năng cấp dòng trực tiếp từng chân I/O của vi điều khiển. Về cấu tạo, rơ-le gồm một nam châm điện và hệ thống tiếp điểm cơ khí. Khi có dòng chạy qua cuộn dây, lực từ sinh ra sẽ hút cần gạt, làm đóng hoặc mở các tiếp điểm, nhờ đó ta có thể dùng tín hiệu điều khiển điện áp thấp để đóng cắt mạch điện cao hoặc dòng lớn hơn một cách gián tiếp, đồng thời tạo lớp cách ly an toàn giữa khối điều khiển và tải.

LED

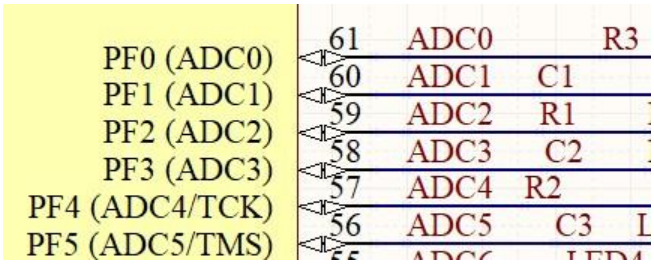
LED là đi-ốt phát quang, viết tắt của "Light Emitting Diode" phát sáng khi được phân cực thuận. Điện áp rơi thuận của LED thường trong khoảng 1,5-3,3 V, dòng làm việc điển hình 10-30 mA tùy loại và màu. Trên bảng mạch thực hành, LED được dùng chủ yếu với vai trò đèn báo trạng thái, đèn nền của các màn hình hiển thị để hỗ trợ người dùng đọc được thông tin trong điều kiện thiếu ánh sáng.

IV. Nội dung bài thực hành

Viết chương trình nhập mật khẩu dạng *** lên màn hình LCD, nếu nhập đúng mật khẩu là số "8888" thì bật role 1, nếu nhập sai mật khẩu thì đèn đỏ sẽ nhấp nháy cảnh báo, màn hình sẽ yêu cầu nhập lại mật khẩu, sau 3 lần nhập sai mật khẩu sẽ bật role 2.

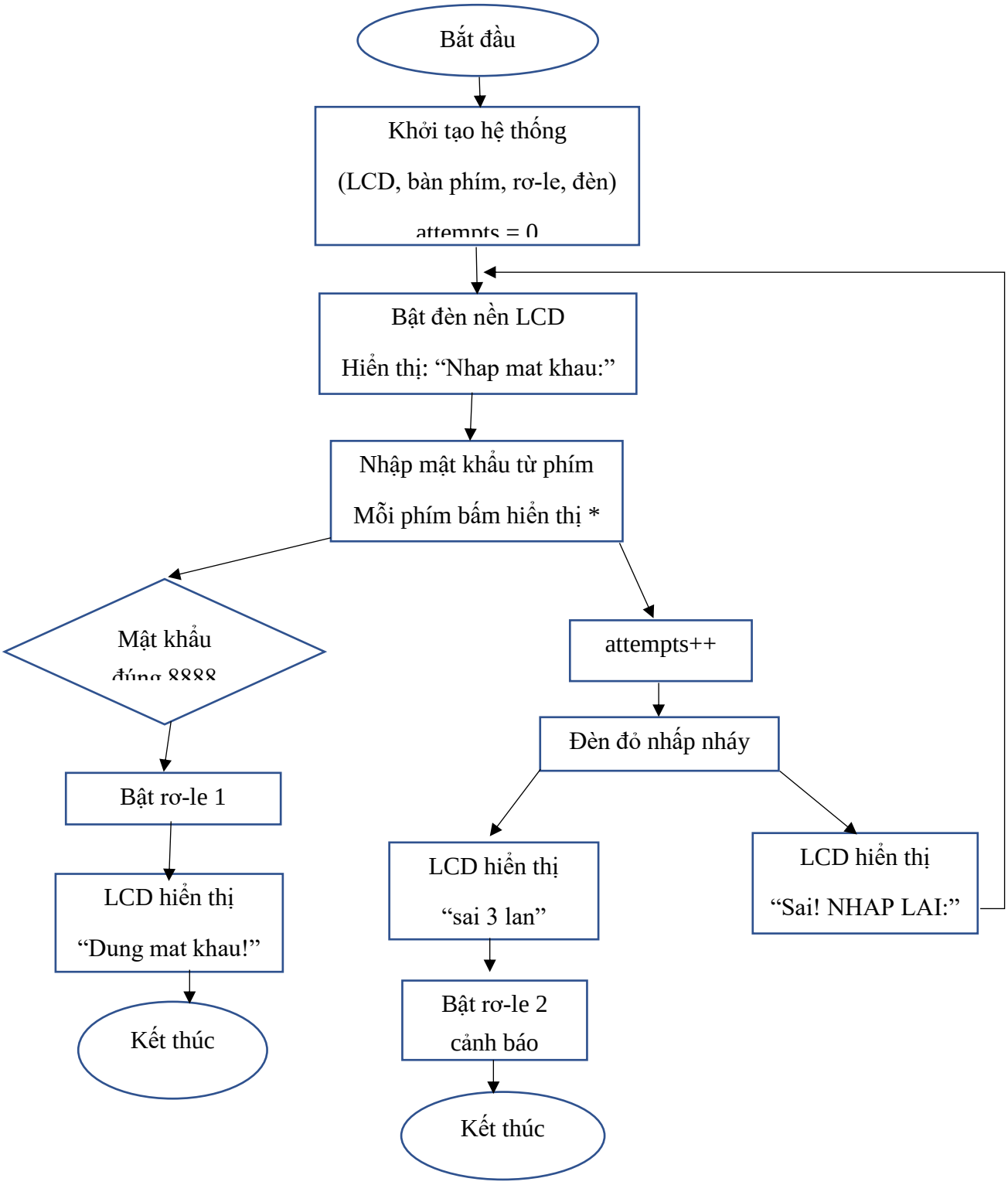
1. Sơ đồ kết nối

Trong bài thực hành này, keypad 3x3 được nối với vi xử lý Atmega128 như ở Hình 6.3.



Hình 6.3. Kết nối keypad 3x3 với vi xử lý Atmega128

2. Sơ đồ thuật toán

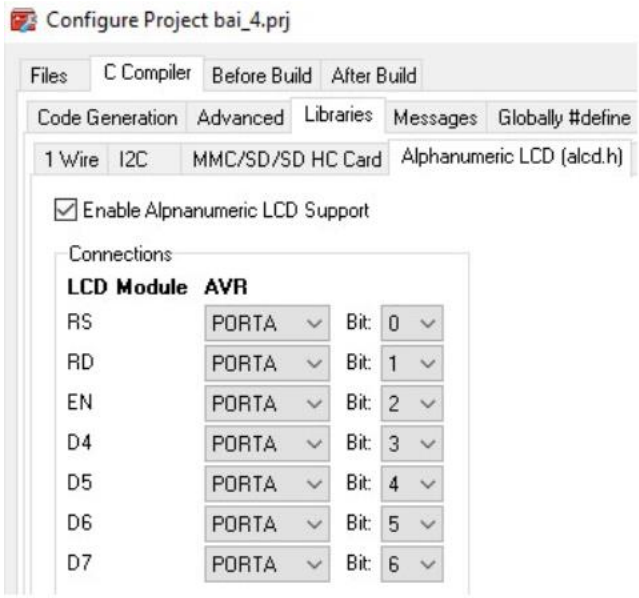


3. Mã chương trình

Chúng ta tạo một thư mục có tên BAI 6, vào CodeVison tạo một project “bai 6” lưu vào trong thư mục vừa tạo, đồng thời cấu hình LCD theo thứ tự đường dẫn:

Project> Configure > C Compiler > Libraries > Alphanumeric LCD

Khi đó một bảng cấu hình sẽ hiện ra, chúng ta sẽ điền các thông tin sau như ở Hình 6.5.



Hình 6.5. Cấu hình để sử dụng màn hình LCD 16x2 trong CodeVision

Trong cửa sổ soạn thảo, chúng ta viết đoạn mã sau:

tệp tin có tên “bai 6.c” có chứa đoạn mã sau bằng trình soạn thảo Nano:

```
#include <io.h>
#include <delay.h>
#include <alcd.h>
int  keypad[3][3] = {0,1,2,3,4,5,6,7,8};
char pass[5]; // tao mang ki tu luu tru mat khau
int  idx = 0; //  bien dem so ki tu
int  attempts = 0; // tao bien dem khi nhap sai mk
// ham hien thi loi nhap mk
void hien_thi_loi_nhap(void)
{
    int k; // bien k
    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_puts("Nhap MK:"); // hien thi NHAP MK tai hang 1
    lcd_gotoxy(0,1); // hien thi tai hang 2
```

```
16      for (k=0; k<idx; k++) lcd_putchar('*'); // hien thi ki tu * tuong ung voi so da
    nhap
17  }
18  // hien thi khi nhap dung mat khau
19  void dung_mat_khau(void)
20  {
21      lcd_clear(); // xoa toan bo LCD
22      lcd_gotoxy(0,0);
23      lcd_puts("Dung mat khau!"); // hien thi
24      lcd_gotoxy(0,1);
25      lcd_puts("Relay1 ON");  // hien thi
26  }
27  // hien thi khi nhap sai mat khau
28  void sai_mat_khau(void)
29  {
30      lcd_clear();// xoa toan bo LCD
31      lcd_gotoxy(0,0);
32      lcd_puts("Sai! NHAP LAI:");  //hien thi
33      lcd_gotoxy(0,1);
34      lcd_puts("");  // hien thi mat khau
35  }
36  // hien thi khi nhap sai mk qua 3 lan
37  void khoa_he_thong(void)
38  {
39      lcd_clear();  // xoa toan bo LCD
40      lcd_gotoxy(0,0);
41      lcd_puts("Sai 3 lan!"); // hien thi
42      lcd_gotoxy(0,1);
43      lcd_puts("Relay2 ON"); // hien thi
44  }
45  void xu_ly_phim(int value)
46  {
```

```
47      int t;
48      if (idx < 4) { // khi chuoi mk chua du 4 ki tu
49          pass[idx] = (char)('0' + value);   // ghi so da nhap vao mang pass(MK)
50          idx++;    // so ki tu mk tang them 1
51          lcd_gotoxy(idx-1,1);   // in ki tu tai hang 1 cot do
52          lcd_putchar('*');      // hien thi gia tri *
53          delay_ms(500);         // chong doi
54      }
55      if (idx == 4) {    // khi chuoi mk du 4 ki tu
56          pass[4] = '\0'; // them ki tu ket thuc
57          if (pass[0]=='8' && pass[1]=='8' && pass[2]=='8' && pass[3]=='8')
    { //kiem tra mk
58              PORTC.3 = 1;      // mat khau dung, relay1 bat
59              PORTD.4 = 0;      // LED tat
60              dung_mat_khau();  // hien thi ra man hinh
61              while(1){}
62          } else {  // mat khau sai
63              // LED do nhay 3 lan
64              for (t=0; t<3; t++){
65              PORTD.4 = 1;
66              delay_ms(150);
67              PORTD.4 = 0;
68              delay_ms(150); }
69              attempts++; // so lan nhap mk tang them 1
70              idx = 0;  // so ki tu mk reset ve 0
71              pass[0]=pass[1]=pass[2]=pass[3]='\0'; // xoa noi dung 4 ki tu vua nhap
72              // kiem tra so lan nhap mk sai
73              if (attempts >= 3) { // so lan nhap mk sai >=3
74                  PORTC.2 = 1;   //relay 2 bat
75                  khoa_he_thong();// hien thi
76                  while(1){}  // vong lap mai mai neu dk dung
77              } else {  // so lan nhap sai mk <3
```

```
78                  sai_mat_khau();  // hien thi
79              }
80          }
81      }
82  }
83  void BUTTON(void)
84  {
85      char a; // tao bien a kiem tra trang thai button
86      int i, j;// hang i cot j
87  for (j=0; j<3; j++) {
88          if (j==0) PORTF = 0b11111101;      // C1=0 muc thap (cot 1)
89          if (j==1) PORTF = 0b11110111;      // C2=0 muc thap (cot 2)
90          if (j==2) PORTF = 0b11011111;      // C3=0 muc thap  (cot 3)
91          for (i=0; i<3; i++) {
92              if (i==0) {                 // R1 = PF2
93                  a = PINF & 0x04;   // trang thai chua duoc nhan (=1)
94                  // neu trang thai duoc nhan(=0), so duoc nhan tai hang cot do
95                  if (a != 0x04) { xu_ly_phim(keypad[i][j]); }
96              }
97              if (i==1) {                 // R2 = PF4
98                  a = PINF & 0x10;
99                  if (a != 0x10) { xu_ly_phim(keypad[i][j]); }
100             }
101             if (i==2) {                 // R3 = PF0
102                 a = PINF & 0x01;
103                 if (a != 0x01) { xu_ly_phim(keypad[i][j]); }
104             }
105         }
106     }
107 }
108 void main(void)
109 {
```

```
110         // LCD
111         lcd_init(16);
112         DDRF =  0b11101010; // C1,C2,C3 o trang thai output, R1,R2,R3 trang thai
        input
113         PORTF = 0b00010101; // keo R1,R2,R3 pull up len 1
114         // LED d? & Relay
115         DDRD.4 = 1;  PORTD.4 = 0;            // LED do tat
116         DDRC.3 = 1;  PORTC.3 = 0;            // Relay1 tat
117         DDRC.2 = 1;  PORTC.2 = 0;            // Relay2 tat
118         // ham cho nhap mk
119         idx = 0;
120         hien_thi_loi_nhap();
121         while(1) {
122            BUTTON(); // nhap mk
123         }
124     }
```

4. Hướng dẫn thực hành

Trong đoạn chương trình trên, các chân của keypad 3x3 được kết nối với các GPIO của vi xử lý Atmega128 theo hình 6.3, do đó để giao tiếp, chúng ta phải cấu hình trong chương trình CodeVision theo hình 6.4.

Sau khi soạn thảo chương trình xong, chúng ta biên dịch chương trình để tạo mã hex, tiếp tục sử dụng chương trình Progisp để burn mã hex của chương trình trên vào chíp và quan sát trên màn hình LCD.

Tiến hành test các mật khẩu sai, mật khẩu đúng để xem hiển thị trên LCD, đèn led và rơle.

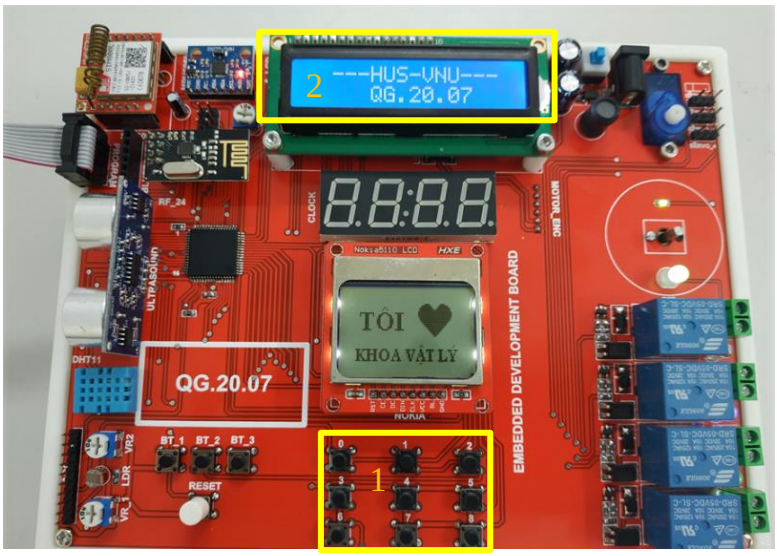
BÀI 7. GIAO TIẾP BÀN PHÍM MA TRẬN, LCD 16x2

I. Mục đích

- Giao tiếp với bàn phím ma trận và màn hình LCD với Atmega128.
- Hiển thị văn bản lên màn hình LCD 16x2.

II. Phần cứng chủ yếu sử dụng trong bài thực hành

Trong bài thực hành này, chúng ta sử dụng các nút bấm của bàn phím ma trận, hiển thị tín hiệu với màn hình LCD. Các phần cứng này có sẵn trên bảng mạch thực hành ở các vị trí 1, 2 trên Hình 7.1.

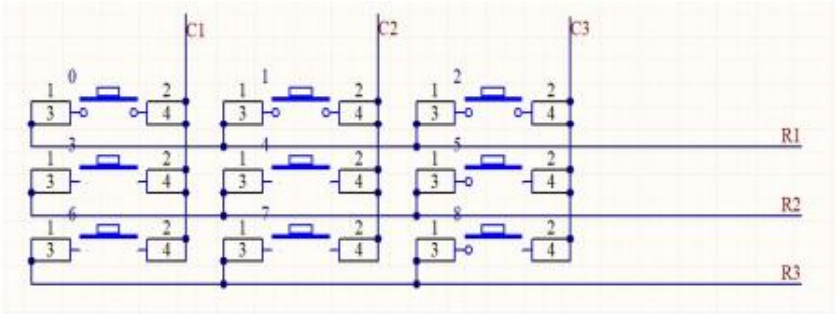


Hình 7.1. Phần cứng sử dụng trong bài thực hành 7

III. Tóm tắt nội dung lý thuyết

Keypad 3x3

Bàn phím ma trận được tạo thành từ các nút được sắp xếp theo hàng và cột. Mỗi nút nằm chính xác tại vị trí giao nhau của hàng và cột.



Hình 7.2. Cách các dây bên trong bàn phím ma trận 3×3 kết nối từng hàng với từng cột.

R là viết tắt của Row (Hàng); R1, R2, R3 là các chân đại diện cho Hàng 1, Hàng 2, Hàng 3 của bàn phím.

C là viết tắt của Column (Cột); C1, C2, C3 là các chân đại diện cho Cột 1, Cột 2, Cột 3 của bàn phím.

Mỗi nút nhấn trên bàn phím ma trận được đặt tại giao điểm của một Hàng và một Cột.

Khi một nút nhấn được bấm, nó sẽ tạo kết nối điện giữa chân Hàng tương ứng và chân Cột tương ứng.

Chân R1, R2, R3, C1, C2, C3 được kết nối với các chân GPIO (General Purpose Input/Output) của vi điều khiển để thực hiện quá trình quét phím (keypad scanning):

Quét Hàng (Outputs): Vi điều khiển lần lượt đặt trạng thái LOW (hoặc HIGH, tùy thuộc vào cách mắc) lên từng chân R (R1, R2, R3...) trong khi các chân Hàng khác giữ ở trạng thái HIGH (hoặc Z/input). Các chân R hoạt động như Đầu ra (Output).

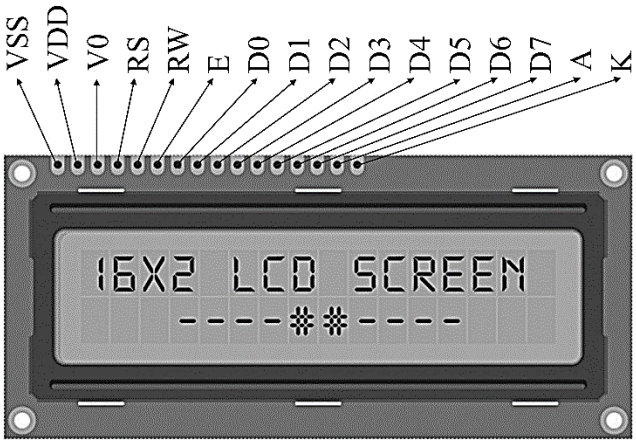
Đọc Cột (Inputs): Vi điều khiển đọc trạng thái của các chân C (C1, C2, C3...). Các chân C thường được cấu hình với trở kéo lên (pull-up resistors) và hoạt động như Đầu vào (Input).

Xác định phím: Nếu một phím được bấm, tín hiệu LOW từ chân Hàng đang được kích hoạt sẽ truyền qua nút nhấn và làm cho chân Cột tương ứng chuyển sang trạng thái LOW (hoặc ngược lại).

Bằng cách biết Hàng nào đang được kích hoạt (R1, R2 hay R3...) và Cột nào nhận được tín hiệu thay đổi (C1, C2 hay C3...), vi điều khiển sẽ xác định được chính xác phím nào đã được nhấn.

* ***Màn hình LCD 16x2***

Màn hình LCD 16x2 là một thiết bị ngoại vi dùng để giao tiếp với người dùng một cách trực quan dựa trên công nghệ tinh thể lỏng. Các tinh thể lỏng trong màn hình được kích thích bởi các tín hiệu điện, từ đó thay đổi cách chúng phản xạ ánh sáng để tạo ra các ký tự và hình ảnh trên màn hình. Màn hình có khả năng hiển thị 16 ký tự trên mỗi dòng và tổng cộng 2 dòng, có tích hợp đèn nền giúp đọc thông tin dễ dàng hơn trong điều kiện ánh sáng yếu. Khi sản xuất LCD 16x2, nhà sản xuất đã tích hợp chíp điều khiển loại 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 Hình 7.3.



Hình 7.3. Giao diện của màn hình LCD 16x2 thông thường

Chức năng các chân của màn hình LCD 16x2 được mô tả như ở Bảng 7.1 dưới đây.

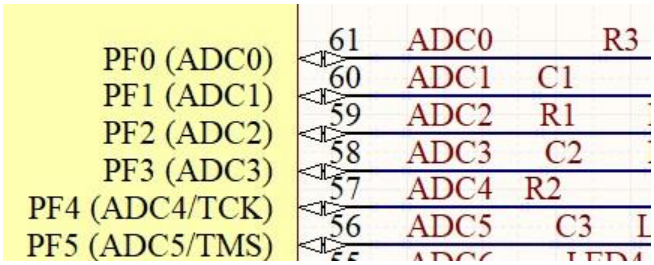
Bảng 7.1. Chức năng của các chân màn hình LCD 16x2

IV. Nội dung bài thực hành

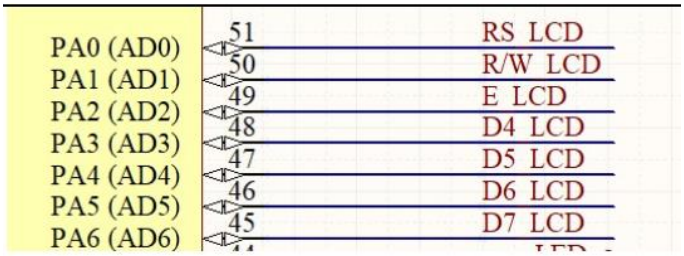
Viết chương trình quét ma trận phím 3x3 và hiển thị số thứ tự phím lên LCD.

1. Sơ đồ kết nối

Trong bài thực hành này, keypad 3x3 và LCD 16x2 được nối với vi xử lý Atmega128 như ở Hình 7.4 và 7.5.

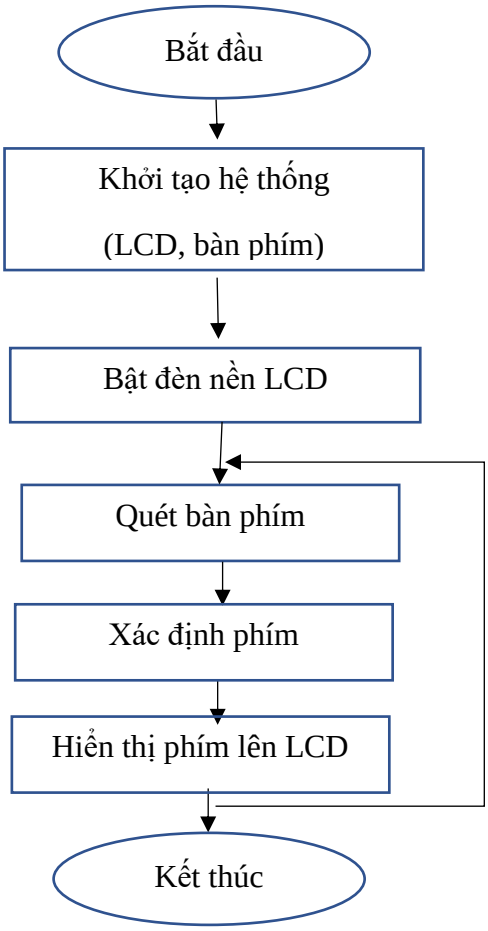


Hình 7.4. Kết nối keypad 3x3 với vi xử lý Atmega128



Hình 7.5. Kết nối LCD 16x2 với vi xử lý Atmega128

2. Sơ đồ thuật toán

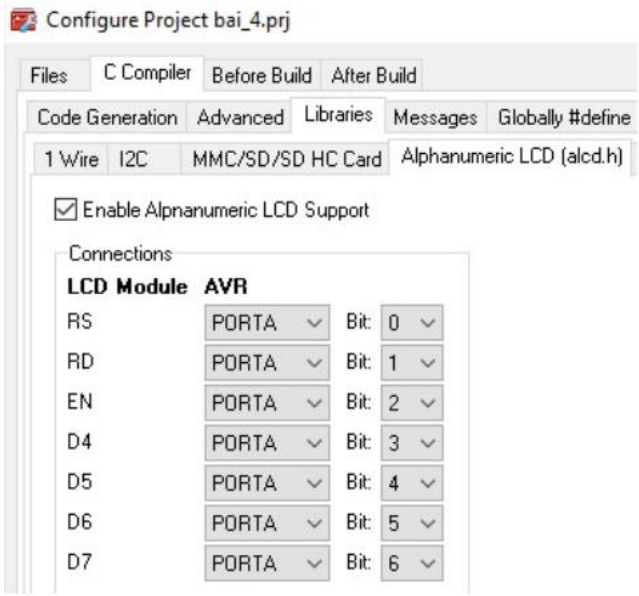


3. Mã chương trình

Chúng ta tạo một thư mục có tên BAI 7, vào CodeVison tạo một project “bai 7” lưu vào trong thư mục vừa tạo, đồng thời cấu hình LCD theo thứ tự đường dẫn:

Project> Configure > C Compiler > Libraries > Alphanumeric LCD

Khi đó một bảng cấu hình sẽ hiện ra, chúng ta sẽ điền các thông tin sau như ở Hình 7.7.



Trong cửa sổ soạn thảo, chúng ta viết đoạn mã sau:

tệp tin có tên “bai 7.c” có chứa đoạn mã sau bằng trình soạn thảo Nano:

```
1   #include <io.h>
2   #include <delay.h>
3   #include <stdio.h>
4   #include <stdint.h>
5   #include <stdlib.h>
6   #include <alcd.h>
7   int keypad[3][3] = {1,2,3,4,5,6,7,8,9};
8   char data[16];
9   void BUTTON() {
10      char a;
11      int i,j;
12      for(j=0; j<3; j++){
13          if(j == 0)
14              PORTF = 0b11111101;
15          if (j == 1)
16              PORTF = 0b11110111;
```

```
17          if (j == 2)
18              PORTF = 0b11011111;
19              for (i=0; i<3; i++) {
20                  if (i == 0) {
21                      a = PINF&0x04;
22                      if( a!=0x04){
23                      lcd_gotoxy(0,0);
24                      sprintf(data, "%u", keypad[i][j]);
25                      lcd_puts(data);
26                      delay_ms(500);
27                      }}
28                  if (i == 1) {
29                      a = PINF&0x10;
30                      if( a!=0x10) {
31                      lcd_gotoxy(0,0);
32                      sprintf(data, "%u", keypad[i][j]);
33                      lcd_puts(data);
34                      delay_ms(500);
35                      }}
36                  if (i == 2) {
37                      a = PINF&0x01;
38                      if (a != 0x01){
39                      lcd_gotoxy(0,0);
40                      sprintf(data, "%u", keypad[i][j]);
41                      lcd_puts(data);
42                      delay_ms(500);
43                      }}
```

```
44        }
45        }
46        }
47    void main()
48    {
49        DDRD = 0xFF;
50        PORTD = 0xB6;
51        lcd_init(16);
52        DDRF = 0b11101010;
53        PORTF = 0b00010101;
54    while (1)
55        {
56        BUTTON();
```

4. Hướng dẫn thực hành

Trong đoạn chương trình trên, các chân của keypad 3x3 được kết nối với các GPIO của vi xử lý Atmega128 theo hình 7.4, do đó để giao tiếp, chúng ta phải cấu hình trong chương trình CodeVision theo hình 7.7.

Sau khi soạn thảo chương trình xong, chúng ta biên dịch chương trình để tạo mã hex, tiếp tục sử dụng chương trình Progisp để burn mã hex của chương trình trên vào chíp và quan sát trên màn hình LCD.

Tiến hành quét các phím để xem hiển thị trên LCD.

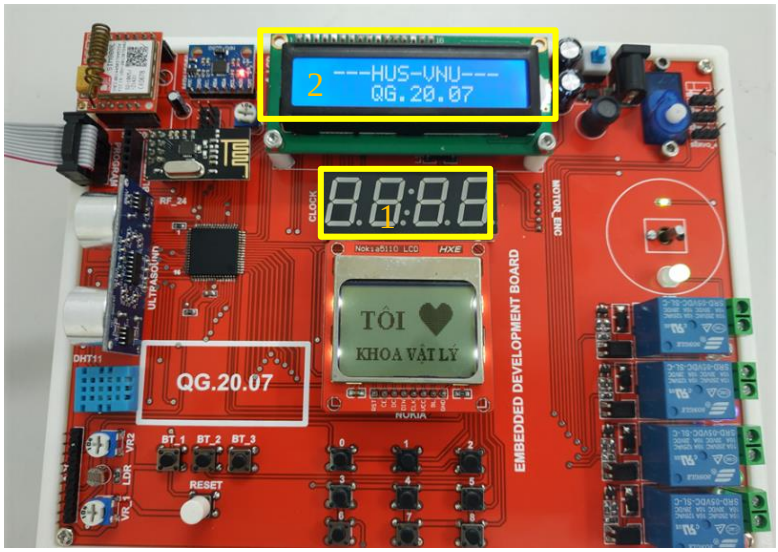
BÀI 8. GIAO TIẾP LED 7 ĐOẠN

I. Mục đích

- Giao tiếp với bàn phím ma trận và màn hình LCD với Atmega128.
- Hiển thị văn bản lên màn hình LCD 16x2.

II. Phần cứng chủ yếu sử dụng trong bài thực hành

Trong bài thực hành này, chúng ta sử dụng hiển thị văn bản với led 7 đoạn và màn hình LCD. Các phần cứng này có sẵn trên bảng mạch thực hành ở các vị trí 1, 2 trên Hình 8.1.



Hình 8.1. Phần cứng sử dụng trong bài thực hành 8

III. Tóm tắt nội dung lý thuyết

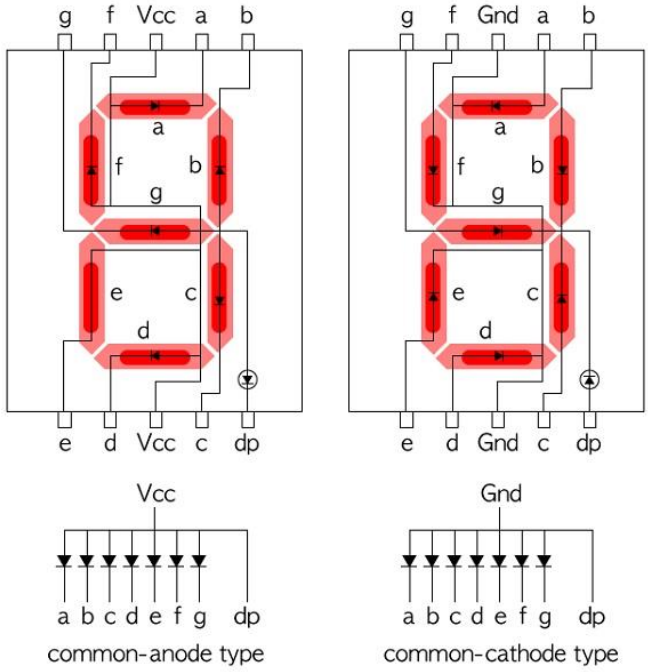
Led 7 đoạn hay còn gọi là "seven segment display" là một module hiện thị kỹ thuật số có thiết kế đặc biệt để hiển thị các thông tin về số. Led 7 đoạn được được sắp xếp theo hình số 8 với 7 đoạn led, mỗi led là một đoan và khi được chiếu sáng sẽ trở thành một phần của một của một chữ số. Led thứ 8 thường là dấu chấm thập phân, để hiển thị các số ≥10 cần ghép từ 2 module trở lên. Các số được tạo ra bằng các bật/tắt các thành led để tạo thành.

Theo phân loại thì LED 7 đoạn có hai loại phổ biến như sau:

LED 7 đoạn Cathode chung (Cực âm chung): tức là các diod có cực âm được nối chung với nhau. Nghĩa là đối với loại led 7 đoạn Cathode chung này, nếu bạn muốn

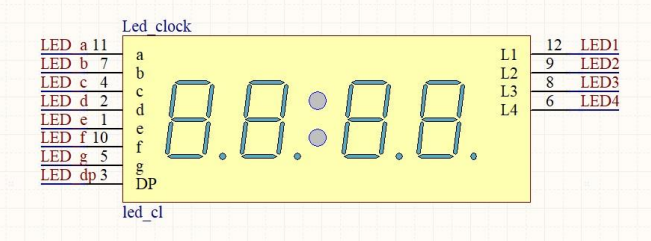
đoạn led nào sáng thì bạn cấp nguồn dương cho chân đoạn đó (+3V, +5V, +12V hay +24V hoặc tùy loại) và cấp nguồn âm hoặc 0V vào chân chung.

LED 7 đoạn Anode chung (Cực dương chung): cách phân cực cho LED ngược lại so với Cathode chung. Tức là tất cả cực dương của các diode của các đoạn LED sẽ được nối chung. Nghĩa là đối với Led 7 đoạn Anode chung, muốn đoạn led nào sáng thì cấp nguồn dương vào chân chung và cấp nguồn âm hoặc 0V vào chân đoạn led đó. LED 7 đoạn Anode chung được sử dụng nhiều hơn trong đời sống vì các linh kiện trong mạch điện thường nối nguồn chung, việc điều khiển sẽ trở nên đơn giản hơn và thiết kế bảng mạch bớt cồng kềnh hơn..



Hình 8.2. Các loại led 7 đoạn

Trong bài thực hành chúng ta không điều khiển LED đơn mà là tổ hợp 4 LED, để hiển thị chúng ta sẽ sử dụng phương pháp quét (bằng việc sử dụng hạn chế của mắt người chỉ có thể phân biệt được với những thay đổi ở khoảng tần số 24 Hz).



Hình 8.3. Sơ đồ các chân của led 7 đoạn

Module LED 7 đoạn bốn chữ số, các chân được chia thành hai nhóm chính: Chân Phân đoạn (Segment Pins) và Chân Chọn Chữ số (Digit Select/Common Pins).

*Chân Phân đoạn (Segment Pins):

Nhóm chân này dùng để điều khiển từng đoạn LED nhỏ (a, b, c, d, e, f, g) và dấu chấm thập phân (DP) để tạo thành các chữ số.

Ký hiệu	**Số chân**	**Chức năng**	**Mô tả**
LED a	11	Điều khiển đoạn **a**	Đoạn trên cùng.
LED b	7	Điều khiển đoạn **b**	Đoạn trên bên phải.
LED c	4	Điều khiển đoạn **c**	Đoạn dưới bên phải.
LED d	2	Điều khiển đoạn **d**	Đoạn dưới cùng.
LED e	1	Điều khiển đoạn **e**	Đoạn dưới bên trái.
LED f	10	Điều khiển đoạn **f**	Đoạn trên bên trái.
LED g	5	Điều khiển đoạn **g**	Đoạn giữa.
LED dp	3	Điều khiển dấu **DP**	Dấu chấm thập phân.

*Chân Chọn Chữ số (Digit Select / Common Pins):

Nhóm chân này dùng để chọn chữ số (digit) nào sẽ được hiển thị các phân đoạn (a-g, dp) đã được kích hoạt. Điều này cho phép module 4 chữ số hiển thị các số khác nhau trên mỗi chữ số thông qua kỹ thuật quét LED (Multiplexing).

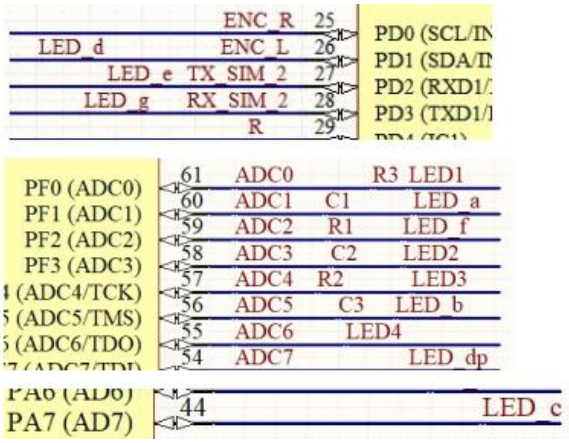
Ký hiệu	**Số chân**	**Chức năng**	**Mô tả**
LED1	12	Chân **Chung (Common)** cho chữ số thứ nhất (bên trái).	Điều khiển BẬT/TẮT chữ số 1.
LED2	9	Chân **Chung (Common)** cho chữ số thứ hai.	Điều khiển BẬT/TẮT chữ số 2.
LED3	8	Chân **Chung (Common)** cho chữ số thứ ba.	Điều khiển BẬT/TẮT chữ số 3.
LED4	6	Chân **Chung (Common)** cho chữ số thứ tư (bên phải).	Điều khiển BẬT/TẮT chữ số 4.

IV. Nội dung bài thực hành

Viết chương trình đếm và hiển thị các số từ 11,00 đến số 99,99, các số sau dấu phẩy thay đổi theo chu kỳ 0,1s.

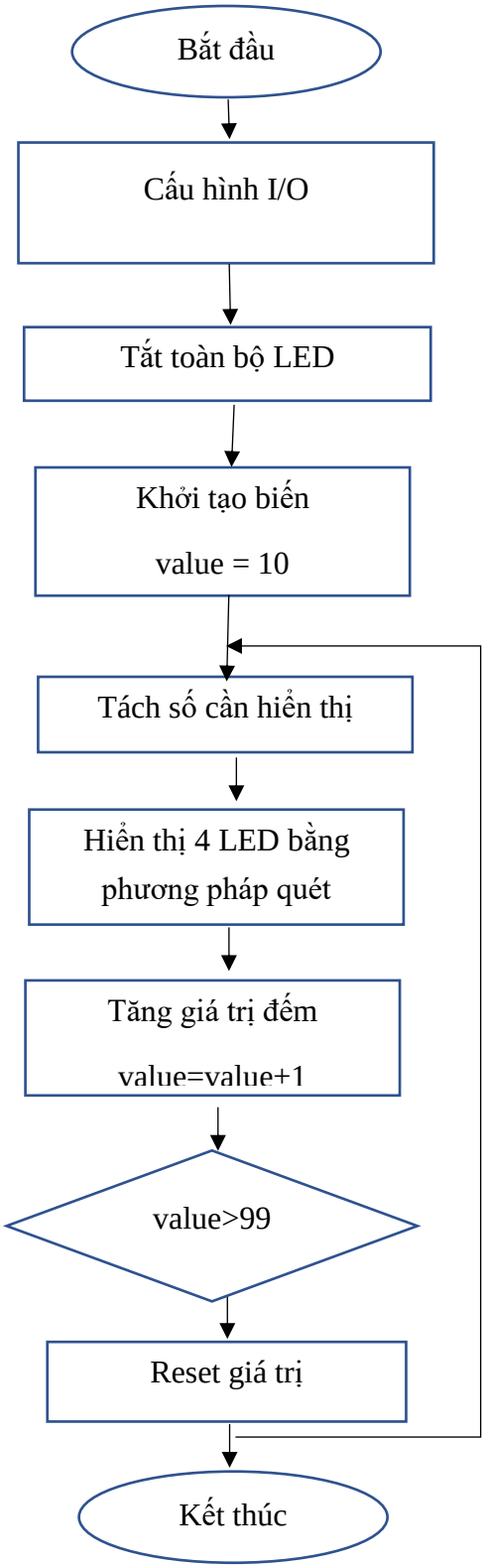
1. Sơ đồ kết nối

Trong bài thực hành này, led 7 đoạn được nối với vi xử lý Atmega128 như ở Hình 8.4.



Hình 8.4. Kết nối 4 LED với vi xử lý Atmega128

2. Sơ đồ thuật toán


Bắt đầu
Cấu hình I/O
Tắt toàn bộ LED
Khởi tạo biến
value = 10
Tách số cần hiển thị
Hiển thị 4 LED bằng
phương pháp quét
Tăng giá trị đếm
value=value+1
value>99
Reset giá trị
Kết thúc

3. Mã chương trình

```
#include <mega128.h>
#include <delay.h>
void digit_all_off(void)
{
   PORTF =0b00000000;
}
void digit1_on(void) { PORTF |= (1<<0); } // PF0
void digit2_on(void) { PORTF |= (1<<3); } // PF3
void digit3_on(void) { PORTF |= (1<<4); } // PF4
void digit4_on(void) { PORTF |= (1<<6); } // PF6
void segment_a(unsigned char want_on){
    if (want_on) PORTF &= ~(1<<1);
    else PORTF |= (1<<1); }
void segment_b(unsigned char want_on){
    if (want_on) PORTF &= ~(1<<5);
    else PORTF |= (1<<5); }
void segment_c(unsigned char want_on){
    if (want_on) PORTA &= ~(1<<7);
    else PORTA |= (1<<7); }
void segment_d(unsigned char want_on){
    if (want_on) PORTD &= ~(1<<1);
    else PORTD |= (1<<1); }
void segment_e(unsigned char want_on){
    if (want_on) PORTD &= ~(1<<2);
    else PORTD |= (1<<2); }
void segment_f(unsigned char want_on){
    if (want_on) PORTF &= ~(1<<2);
    else PORTF |= (1<<2); }
void segment_g(unsigned char want_on){
    if (want_on) PORTD &= ~(1<<3);
    else PORTD |= (1<<3); }
void segment_dp(unsigned char want_on){
    if (want_on) PORTF &= ~(1<<7);
    else PORTF |= (1<<7); }
void segments_all_off(void)
{
   PORTF |= (1<<1); // a
   PORTF |= (1<<5); // b
   PORTA |= (1<<7); // c
   PORTD |= (1<<1); // d
   PORTD |= (1<<2); // e
   PORTF |= (1<<2); // f
   PORTD |= (1<<3); // g
   PORTF |= (1<<7); // dp
}
void show_one_digit(unsigned char n, unsigned char dp_on)
{
```

```
48      switch(n)
49      {
50        case 0: segment_a(1); segment_b(1); segment_c(1); segment_d(1);
      segment_e(1); segment_f(1); segment_g(0); break;
51        case 1: segment_a(0); segment_b(1); segment_c(1); segment_d(0);
      segment_e(0); segment_f(0); segment_g(0); break;
52        case 2: segment_a(1); segment_b(1); segment_c(0); segment_d(1);
      segment_e(1); segment_f(0); segment_g(1); break;
53        case 3: segment_a(1); segment_b(1); segment_c(1); segment_d(1);
      segment_e(0); segment_f(0); segment_g(1); break;
54        case 4: segment_a(0); segment_b(1); segment_c(1); segment_d(0);
      segment_e(0); segment_f(1); segment_g(1); break;
55        case 5: segment_a(1); segment_b(0); segment_c(1); segment_d(1);
      segment_e(0); segment_f(1); segment_g(1); break;
56        case 6: segment_a(1); segment_b(0); segment_c(1); segment_d(1);
      segment_e(1); segment_f(1); segment_g(1); break;
57        case 7: segment_a(1); segment_b(1); segment_c(1); segment_d(0);
      segment_e(0); segment_f(0); segment_g(0); break;
58        case 8: segment_a(1); segment_b(1); segment_c(1); segment_d(1);
      segment_e(1); segment_f(1); segment_g(1); break;
59        default: // 9
60          segment_a(1);    segment_b(1);    segment_c(1);    segment_d(1);
      segment_e(0); segment_f(1); segment_g(1); break;
61      }
62      if (dp_on) segment_dp(1); else segment_dp(0);
63    }
64    void show_4digits_for_ms(unsigned char d1, unsigned char d2,
65                             unsigned char d3, unsigned char d4,
66                             unsigned int total_ms)
67    {
68      unsigned int t;
69      for (t=0; t<total_ms; t+=4)
70      {
71        digit_all_off();
72        show_one_digit(d1, 0);
73        digit1_on();
74        delay_ms(1);
75        digit_all_off();
76        show_one_digit(d2, 0); // dp t?t
77        digit2_on();
78        delay_ms(1);
79        digit_all_off();
80        show_one_digit(d3, 0);
81        digit3_on();
82        delay_ms(1);
83        digit_all_off();
84        show_one_digit(d4, 0);
85        digit4_on();
```

```
86              delay_ms(1);
87          }
88      }
89      void main(void)
90      {
91          unsigned int value;
92          unsigned char d1, d2, d3, d4;
93          DDRF = 0xFF;
94          DDRA = 0xFF;
95          DDRD = 0xFF;
96          digit_all_off();
97          segments_all_off();
98          value = 10;
99          while (1)
100         {
101             d1 = (value/1000)%10;
102             d2 = (value/100)%10;
103             d3 = (value/10)%10;
104             d4 = value%10;
105             show_4digits_for_ms(d1, d2, d3, d4, 500);
106             value++;
107             if (value > 99) value = 10;
108         }
109     }
```

4. Hướng dẫn thực hành

Trong đoạn chương trình trên, các chân của led 7 đoạn được kết nối với các GPIO của vi xử lý Atmega128 theo hình 8.4, do đó để giao tiếp, chúng ta phải cấu hình trong chương trình CodeVision theo hình 8.6.

Sau khi soạn thảo chương trình xong, chúng ta biên dịch chương trình để tạo mã hex, tiếp tục sử dụng chương trình Progisp để burn mã hex của chương trình trên vào chíp và quan sát trên màn hình LCD.

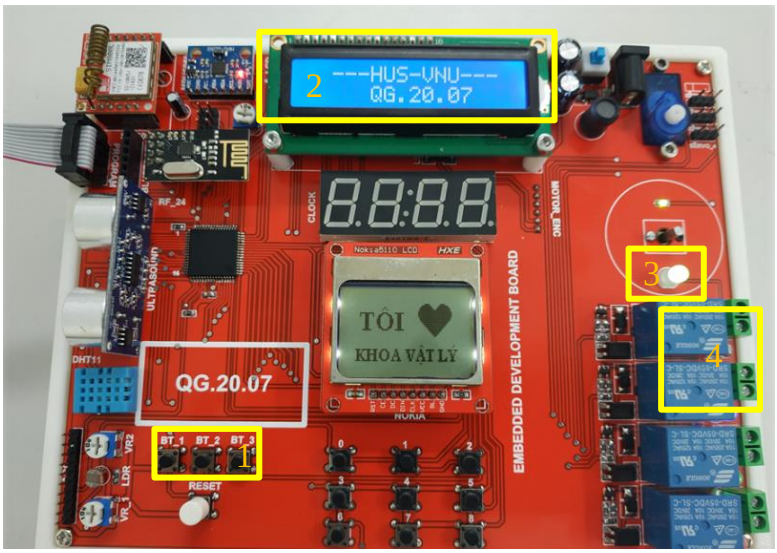
BÀI 9. GIAO TIẾP GPIO + LCD

I. Mục đích

- Giao tiếp với General-Purpose Input/Output và màn hình LCD với Atmega128.
- Hiển thị văn bản lên màn hình LCD 16x2.

II. Phần cứng chủ yếu sử dụng trong bài thực hành

Trong bài thực hành này, chúng ta sử dụng các nút bấm Button của vi điều khiển, hiển thị tín hiệu với màn hình LCD, đèn led RGB và rơle 1,2. Các phần cứng này có sẵn trên bảng mạch thực hành ở các vị trí 1,2,3,4 trên Hình 9.1.



Hình 9.1. Phần cứng sử dụng trong bài thực hành

III. Tóm tắt nội dung lý thuyết

Giao tiếp GPIO

ATmega128 là vi điều khiển 8-bit thuộc họ AVR của Microchip/Atmel. Một trong những khối phần cứng quan trọng nhất của ATmega128 là bộ các cổng vào/ra số (GPIO – General Purpose Input/Output).

Các GPIO được chia thành các port 8 bit: PORTA, PORTB, PORTC, …, PORTG, mỗi chân có thể cấu hình là input hoặc output thông qua các thanh ghi:

DDRn (Data Direction Register): cấu hình hướng chân (1 = output, 0 = input)

PORTn: khi output → ghi mức HIGH/LOW; khi input → bật điện trở kéo lên (pull-up)

PINn: đọc trạng thái logic tại chân

Đặc điểm GPIO của ATmega128:

Điện áp hoạt động: 5V

Dòng tối đa mỗi chân ~20mA

Tốc độ chuyển mạch nhanh → tương thích LCD

Phân nhóm port rõ ràng, dễ mapping cho module LCD

Sử dụng GPIO là nền tảng để điều khiển các thiết bị ngoại vi dạng số như LED, keypad hay LCD ký tự 16x2.

DDRxn	PORTxn	PUD (Trong thanh ghi SFIOR	I/O	Pull-up	**Chú thích**
0	0	x	Ngõ vào	không	Cao trở
0	1	0	Ngõ vào	có	Như một nguồn dòng
0	1	1	Ngõ vào	không	Cao trở
1	0	x	Ngõ ra	không	Ngõ ra thấp
1	1	x	Ngõ ra	không	Ngõ ra cao

Hình 9.2. Cấu hình cho các chân cổng

DDRxn là bit thứ n của thanh ghi DDRx

PORTxn là bit thứ n của thanh ghi PORTx

Dấu x ở cột 3 để chỉ giá trị logic là tùy ý

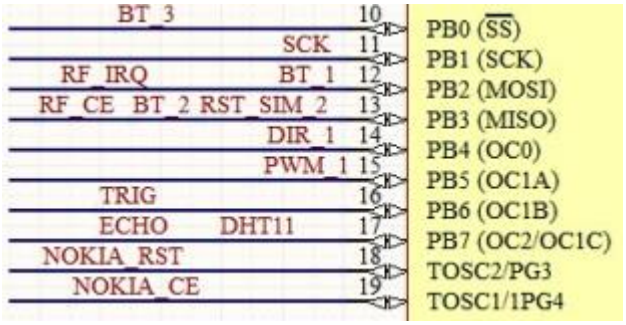
***Nút bấm, LED, rơ-le đã trình bày ở bài 1.**

***Giao tiếp LCD đã trình bày ở bài 7.**

IV. Nội dung bài thực hành

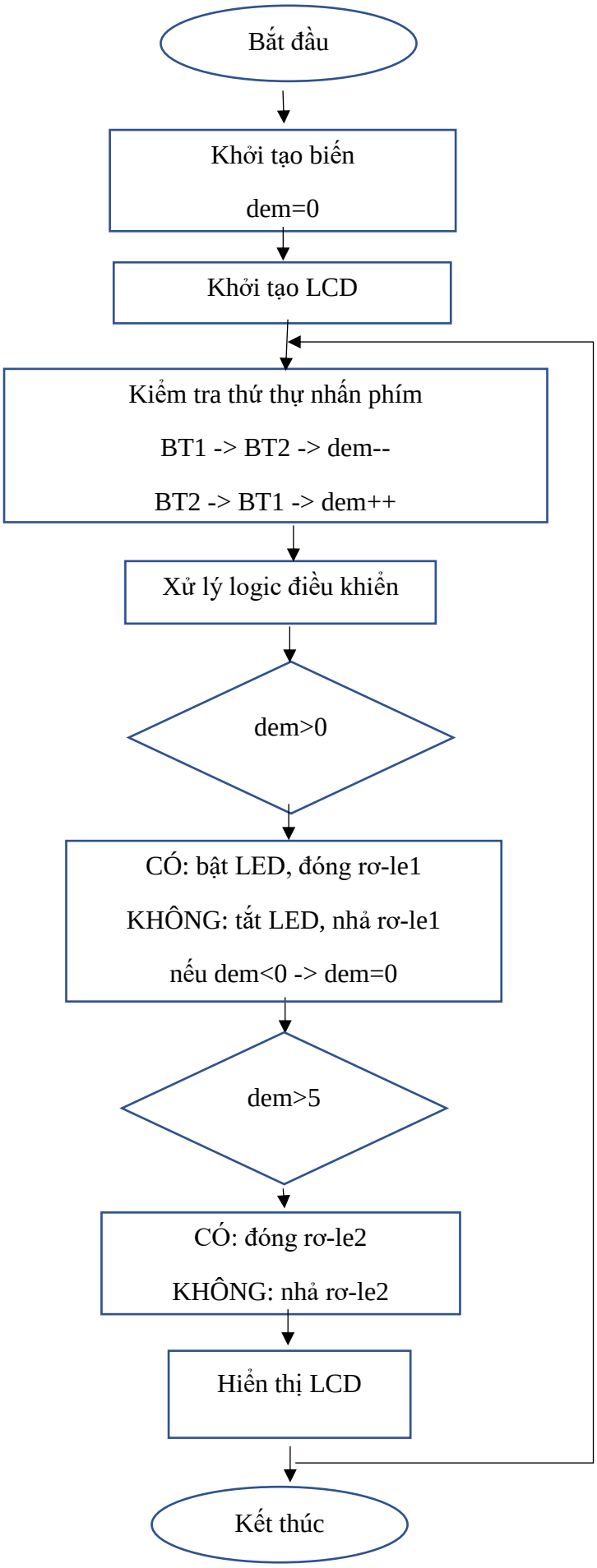
Viết chương trình bấm BT_1 trước, bấm BT_2 sau thì đếm sự kiện tăng dần, nếu bấm BT_2 trước, bấm BT_1 sau thì đếm sự kiện giảm dần, nếu sự kiện lớn hơn 0 thì bật đèn và đóng role 1; nếu sự kiện nhỏ hơn 0 thì xác lập sự kiện bằng 0 và tắt đèn đồng thời nhà role 1. Nếu sự kiện lớn hơn 5 thì bật thêm role 2, sự kiện nhỏ hơn 5 thì nhả role2.

1. Sơ đồ kết nối



Hình 9.3. Kết nối với vi xử lý Atmega128

2. Sơ đồ thuật toán


Bắt đầu
Khởi tạo biến
dem=0
Khởi tạo LCD
Kiểm tra thứ thự nhấn phím
BT1 -> BT2 -> dem--
BT2 -> BT1 -> dem++
Xử lý logic điều khiển
dem>0
CÓ: bật LED, đóng rơ-le1
KHÔNG: tắt LED, nhả rơ-le1
nếu dem<0 -> dem=0
dem>5
CÓ: đóng rơ-le2
KHÔNG: nhả rơ-le2
Hiển thị LCD
Kết thúc

3. Mã chương trình

```
#include <io.h>
#include <alcd.h>
#include <delay.h>
#include <stdlib.h>
#define BT1 PINB.2
#define BT2 PINB.3
#define LED_PIN PORTD5        // Chân D.5 (LED)
#define ROLE1_PIN PORTC3      // Chân C.3 (Relay 1)
#define ROLE2_PIN PORTC2      // Chân C.2 (Relay 2)
#define LCD_BACKLIGHT_PIN PORTD7 // Chân D.7 (Đèn n?n LCD)
int dem = 0;
char buf[10];
unsigned char last_btn_state = 0;
void check_button_sequence(void)
{
   if (BT1 == 0) {
      delay_ms(5);
      if (BT1 == 0) {
         if (last_btn_state == 2) {
            dem--;
            last_btn_state = 0;
         } else if (last_btn_state == 0) {
            last_btn_state = 1;
         }
         while (BT1 == 0);
      }
   }
   if (BT2 == 0) {
      delay_ms(5);
      if (BT2 == 0) {
         if (last_btn_state == 1) {
            dem++;
            last_btn_state = 0;
         } else if (last_btn_state == 0) {
            last_btn_state = 2;
         }
         while (BT2 == 0);
      }
   }
}
void check_control_logic(void)
{
   // --- 1. LED (D.5) và ROLE 1 (C.3) ---
   if (dem > 0) {
      // BAT LED (D.5)
      PORTD |= (1<<LED_PIN);
      // DONG ROLE 1 (C.3) - Active-Low: LOW
```

```
48          PORTC &= ~(1<<ROLE1_PIN);
49      }
50      else {
51          // TAT LED (D.5)
52          PORTD &= ~(1<<LED_PIN);
53          // NHA ROLE 1 (C.3) - Active-Low: HIGH
54          PORTC |= (1<<ROLE1_PIN);
55          if (dem < 0) {
56              dem = 0;
57          }
58      }
59      // --- 2. ROLE 2 (C.2) ---
60      if (dem > 5) {
61          // Đóng Role 2
62          PORTC &= ~(1<<ROLE2_PIN);          // DONG ROLE 2 (Active-Low:
    LOW)
63      }
64      else {
65          // Nh? Role 2
66          PORTC |= (1<<ROLE2_PIN);        // NHA ROLE 2 (Active-Low: HIGH)
67      }
68  }
69  void display_lcd(void)
70  {
71      itoa(dem, buf);
72      lcd_clear();
73      lcd_gotoxy(0, 0);
74      lcd_puts("SU KIEN:");
75      lcd_gotoxy(9, 0);
76      lcd_puts(buf);
77  }
78  void main(void)
79  {
80      DDRB &= ~((1<<DDB2) | (1<<DDB3));
81      PORTB |= (1<<PORTB2) | (1<<PORTB3);
82      DDRD = 0xA0; // D.7 (LCD), D.5 (LED) là Output
83      PORTD = 0xA0;
84      // D.3 (ROLE1) và D.2 (ROLE2) là Output
85      DDRC |= (1<<DDC3) | (1<<DDC2);
86      lcd_init(16);
87      lcd_clear();
88      // ROLE1 (C.3) và ROLE2 (C.2) v? HIGH (Nh? cho Active-Low)
89      PORTC |= (1<<ROLE1_PIN) | (1<<ROLE2_PIN);
90      PORTD &= ~(1<<LED_PIN);
91      PORTD |= (1<<LCD_BACKLIGHT_PIN);
92      display_lcd();
93      while (1)
94      {
```

```
95              check_button_sequence();
96              check_control_logic();
97              display_lcd();
98              delay_ms(100);
99          }
100     }
```

4. Hướng dẫn thực hành

Trong đoạn chương trình trên, kết nối các GPIO theo hình 9.3. Để giao tiếp, chúng ta phải cấu hình trong chương trình CodeVision theo hình 9.5.

Sau khi soạn thảo chương trình xong, chúng ta biên dịch chương trình để tạo mã hex, tiếp tục sử dụng chương trình Progisp để burn mã hex của chương trình trên vào chíp và thao tác bấm BT1, BT2, quan sát trên màn hình LCD, đèn và rơle.

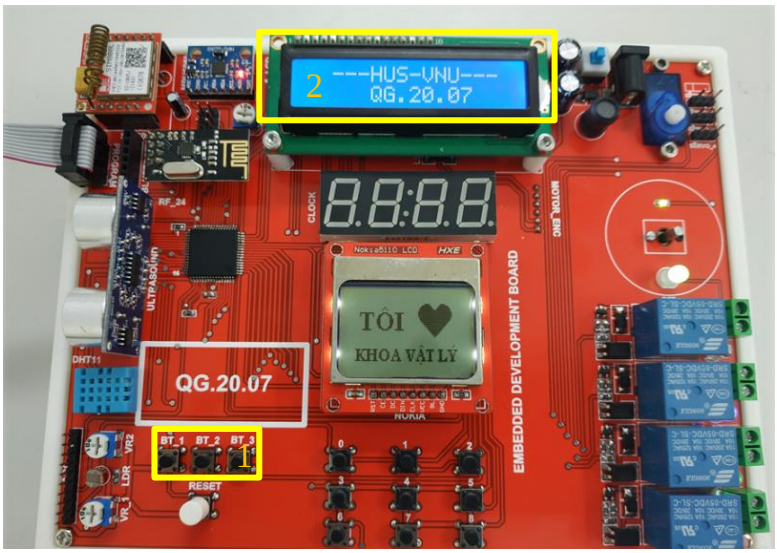
BÀI 10. GIAO TIẾP LCD 16x2, GLCD 84x48

I. Mục đích

- Giao tiếp với màn hình LCD và GLCD với Atmega128.
- Hiển thị văn bản lên màn hình LCD 16x2.

II. Phần cứng chủ yếu sử dụng trong bài thực hành

Trong bài thực hành này, chúng ta sử dụng các nút bấm Button của vi điều khiển, hiển thị tín hiệu với màn hình LCD. Các phần cứng này có sẵn trên bảng mạch thực hành ở các vị trí 1,2 trên Hình 10.1.



Hình 10.1. Phần cứng sử dụng trong bài thực hành

III. Tóm tắt nội dung lý thuyết

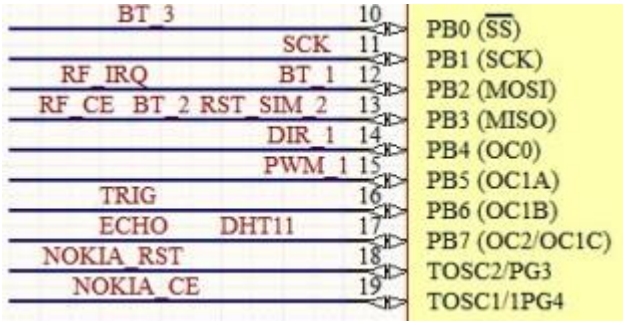
Màn hình LCD đã trình bày ở bài 7.

Nút bấm đã trình bày ở bài 1.

IV. Nội dung bài thực hành

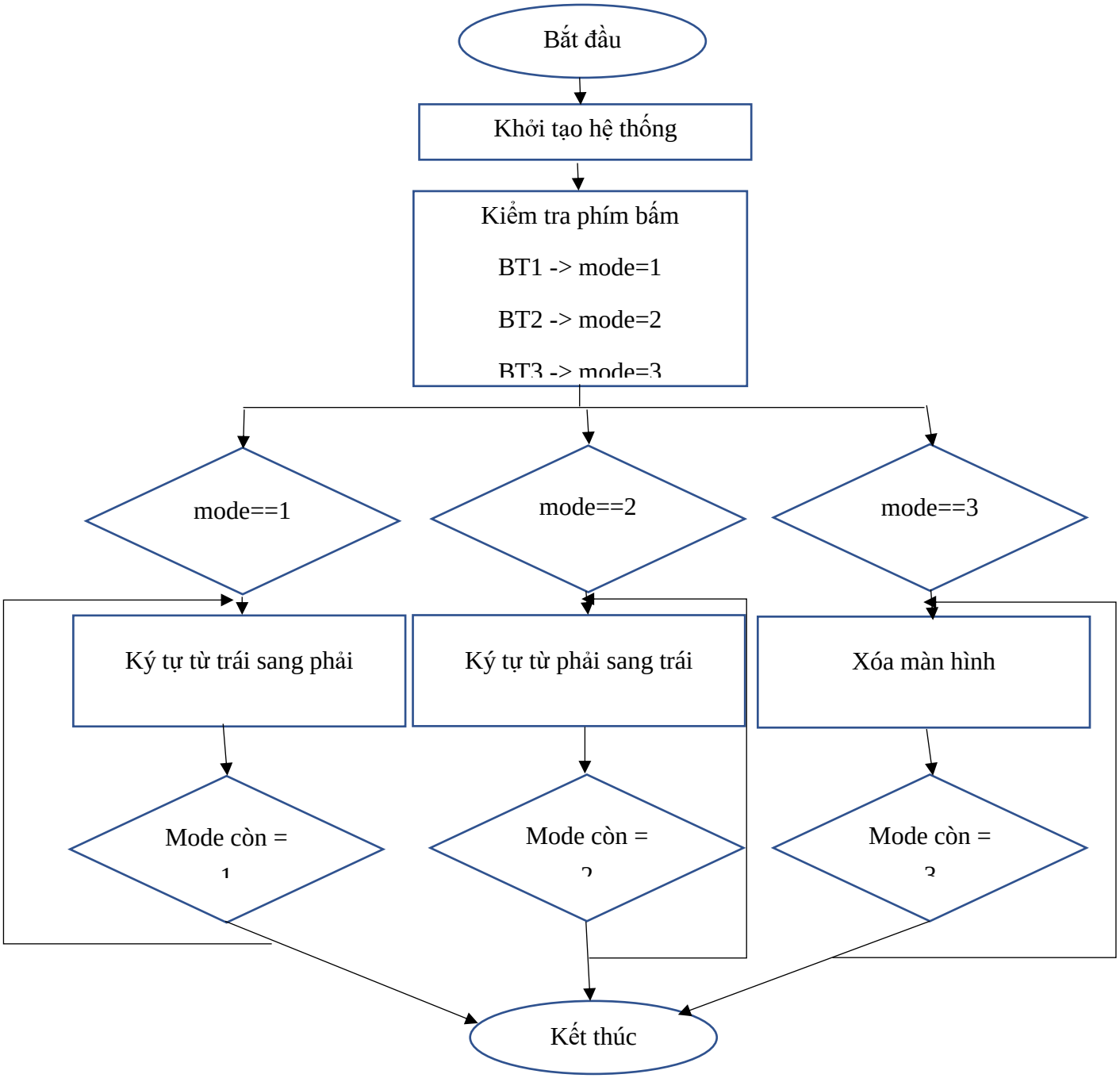
Viết chương trình bấm phím BT1 dòng chữ "hello-world" chạy từng ký tự từ trái sang phải trên màn hình LCD, bấm BT2 thì dòng chữ trên chạy ngược lại, bấm BT3 thì xoá màn hình. (Trong lúc chữ chạy bấm BT bất kỳ thì phải ưu tiên thực hiện nút bấm).

1. Sơ đồ kết nối



Hình 10.2. Kết nối với vi xử lý Atmega128

2. Sơ đồ thuật toán



3. Mã chương trình

```
#include <io.h>
#include <alcd.h>
#include <delay.h>
#define BT1 PINB.2 // Ch?y T->P
#define BT2 PINB.3 // Ch?y P->T
#define BT3 PINB.0 // Xóa màn hình
#define LCD_BT PORTD.7
const flash char message[] = "hello world";
const unsigned char msg_len = 11;
unsigned char mode = 0;
void check_button(void)
{
   if (PINB.2 == 0) { delay_ms(50); if (PINB.2 == 0) { mode = 1; while (PINB.2 == 0); } }
   if (PINB.3 == 0) { delay_ms(50); if (PINB.3 == 0) { mode = 2; while (PINB.3 == 0); } }
   if (PINB.0 == 0) { delay_ms(50); if (PINB.0 == 0) { mode = 3; while (PINB.0 == 0); } }
}
void main(void)
{
   int i;
   signed char k;
   DDRB.2 = 0; PORTB.2 = 1;
   DDRB.3 = 0; PORTB.3 = 1;
   DDRB.0 = 0; PORTB.0 = 1;
   DDRD.7 = 1;
   LCD_BT = 1;
   lcd_init(16);
   lcd_clear();
   while (1)
   {
      check_button();
      if (mode == 1) { // Ch?y T->P (Trái qua Ph?i)
         while (mode == 1) {
            for (k = 0; k < msg_len; k++) { // L?p qua T?NG ký t?
               char current_char = message[k];
               for (i = -1; i <= 16; i++) {
                  lcd_clear();
                  if (i >= 0 && i < 16) {
                     lcd_gotoxy(i, 0); // Luôn in trên Dòng 0
                     lcd_putchar(current_char);
                  }
```

```
                    delay_ms(200);
                    check_button();
                    if (mode != 1) { break; }
                }
                if (mode != 1) { break; }
            }
        }
    } else if (mode == 2) { // Ch?y P->T (Ph?i qua Trái)
        while (mode == 2) {
            for (k = 0; k < msg_len; k++) { // L?p qua T?NG ký t?
                char current_char = message[k];
                for (i = 16; i >= -1; i--) {
                    lcd_clear();
                    if (i >= 0 && i < 16) {
                        lcd_gotoxy(i, 0); // Luôn in trên Dòng 0
                        lcd_putchar(current_char);
                    }
                    delay_ms(200);
                    check_button();
                    if (mode != 2) { break; }
                }
                if (mode != 2) { break; }
            }
        }
    } else if (mode == 3) { // Xóa màn hình
        lcd_clear();
        mode = 0;
    }
  }
}
```

4. Hướng dẫn thực hành

Trong đoạn chương trình trên, kết nối các chân theo hình 10.2. Để giao tiếp, chúng ta phải cấu hình trong chương trình CodeVision theo hình 10.4.

Sau khi soạn thảo chương trình xong, chúng ta biên dịch chương trình để tạo mã hex, tiếp tục sử dụng chương trình Progisp để burn mã hex của chương trình trên vào chíp và thao tác bấm BT1, BT2, BT3 quan sát trên màn hình LCD.

BÀI 11. GIAO TIẾP LCD 16x2, GLCD 84x48

I. Mục đích

- Giao tiếp với màn hình LCD và GLCD với Atmega128.
- Hiển thị văn bản lên màn hình LCD 16x2.

II. Phần cứng chủ yếu sử dụng trong bài thực hành

Trong bài thực hành này, chúng ta sử dụng các nút bấm Button của vi điều khiển, hiển thị tín hiệu với màn hình LCD.

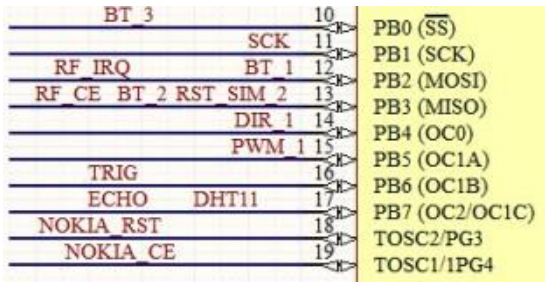
III. Tóm tắt nội dung lý thuyết

Màn hình LCD đã trình bày ở bài 7.

IV. Nội dung bài thực hành

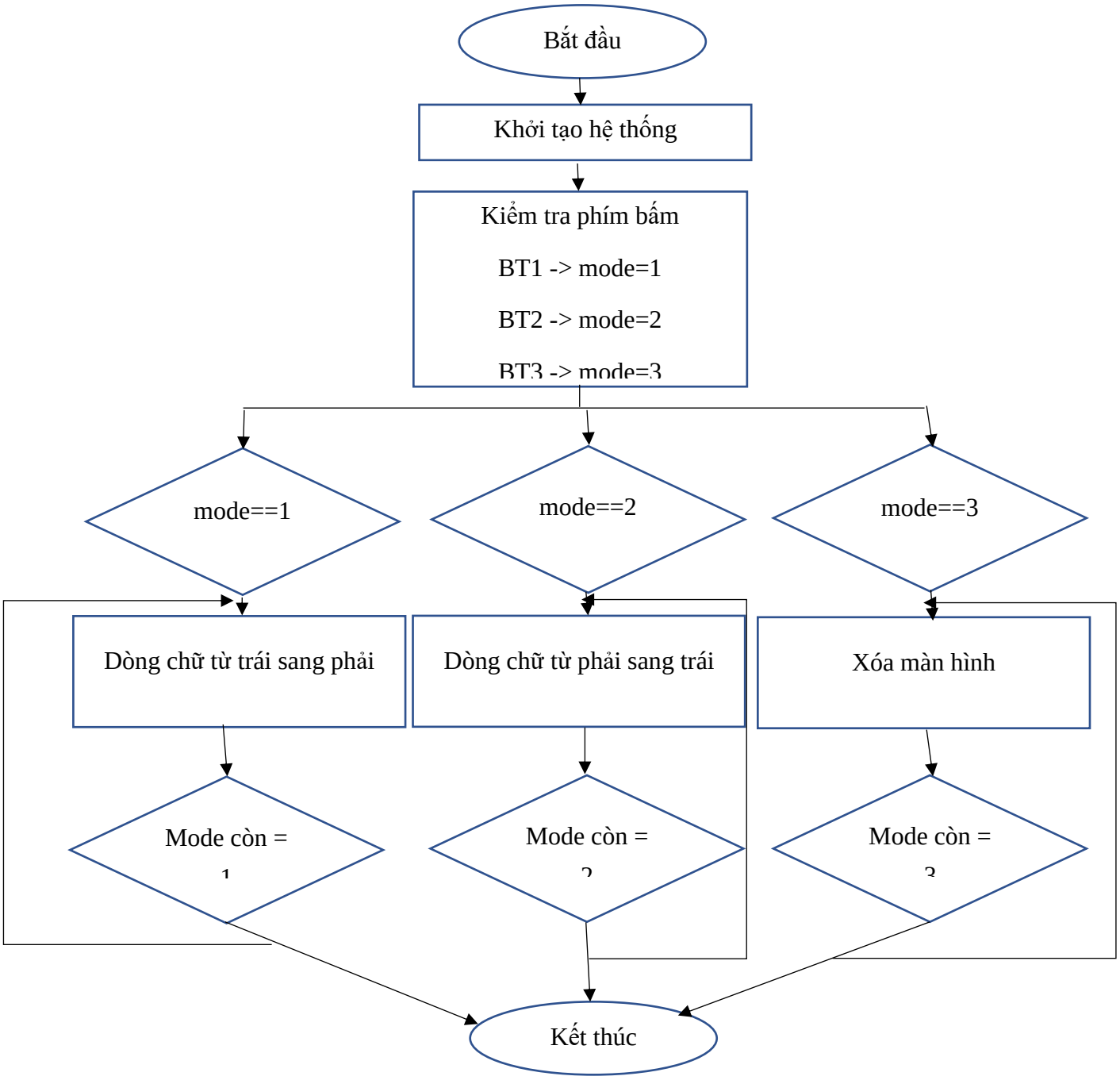
Viết chương trình bấm phím BT 1 dòng chữ "hello-world" chạy từ trái sang phải trên màn hình LCD, bấm BT_2 thì dòng chữ trên chạy ngược lại, bấm BT3 thì xoá màn hình. (Trong lúc chữ chạy bấm BT bất kỳ thì phải ưu tiên thực hiện nút bấm)

1. Sơ đồ kết nối



Hình 11.2. Kết nối với vi xử lý Atmega128

2. Sơ đồ thuật toán


Bắt đầu
Khởi tạo hệ thống
Kiểm tra phím bấm
BT1 -> mode=1
BT2 -> mode=2
BT3 -> mode=3
mode==1
mode==2
mode==3
Dòng chữ từ trái sang phải
Dòng chữ từ phải sang trái
Xóa màn hình
Mode còn =
1
Mode còn =
2
Mode còn =
3
Kết thúc

3. Mã chương trình

```
1   #include <io.h>
2   #include <alcd.h>
3   #include <delay.h>
4   #define BT1 PINB.2
5   #define BT2 PINB.3
6   #define BT3 PINB.0
7   #define LCD_BT PORTD.7
8   const flash char message[] = "hello world";
9   const unsigned char msg_len = 11;
10  unsigned char mode = 0;
11  void check_button(void)
12  {
13     if (BT1 == 0) {
14        delay_ms(50);
15        if (BT1 == 0) {
16           mode = 1;
17           while (BT1 == 0);
18        }
19     }
20     if (BT2 == 0) {
21        delay_ms(50);
22        if (BT2 == 0) {
23           mode = 2;
24           while (BT2 == 0);
25        }
26     }
27     if (BT3 == 0) {
28        delay_ms(50);
29        if (BT3 == 0) {
30           mode = 3;
31           while (BT3 == 0);
32        }
33     }
34  }
35  void main(void)
36  {
37     DDRB.2 = 0; PORTB.2 = 1; // BT1
38     DDRB.3 = 0; PORTB.3 = 1; // BT2
39     DDRB.0 = 0; PORTB.0 = 1; // BT3
40     DDRD.7 = 1;
41     LCD_BT = 1;
42     lcd_init(16);
43     lcd_clear();
44     while (1)
45     {
```

```
46          signed char i;
47          check_button();
48          if (mode == 1) {
49             while (mode == 1) {
50                for (i = 0; i < 16 - msg_len + 1; i++) {
51                   lcd_clear();
52                   lcd_gotoxy(i, 0);
53                   lcd_putsf(message);
54                   delay_ms(200);
55                   check_button();
56                   if (mode != 1) { break; }
57                }
58                if (mode != 1) { break; }
59                for (i = 0; i < 16 - msg_len + 1; i++) {
60                   lcd_clear();
61                   lcd_gotoxy(i, 1);
62                   lcd_putsf(message);
63                   delay_ms(200);
64                   check_button();
65                   if (mode != 1) { break; }
66                }
67                 if (mode != 1) { break; }
68             }
69          } else if (mode == 2) {
70             while (mode == 2) {
71                for (i = 15; i >= -12; i--) {
72                   signed char col_row0 = i + 16;
73                   lcd_clear();
74                   if (i <= 15) {
75                      lcd_gotoxy(i, 1);
76                      lcd_putsf(message);
77                   }
78                   if (col_row0 >= 0 && col_row0 <= 15) {
79                      lcd_gotoxy(col_row0, 0);
80                      lcd_putsf(message);
81                   }
82                   delay_ms(200);
83                   check_button();
84                   if (mode != 2) { break; }
85                }
86             }
87          } else if (mode == 3) { // Xóa màn hình
88             lcd_clear();
89             mode = 0;
90          }
91       }
92    }
```

4. Hướng dẫn thực hành

Trong đoạn chương trình trên, kết nối các chân theo hình 11.2. Để giao tiếp, chúng ta phải cấu hình trong chương trình CodeVision theo hình 11.4.

Sau khi soạn thảo chương trình xong, chúng ta biên dịch chương trình để tạo mã hex, tiếp tục sử dụng chương trình Progisp để burn mã hex của chương trình trên vào chíp và thao tác bấm BT1, BT2, BT3 quan sát trên màn hình LCD.

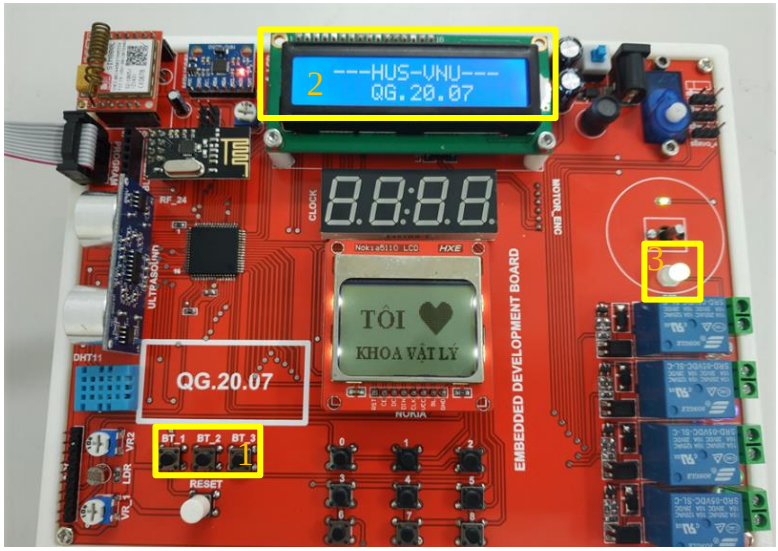
BÀI 12. BỘ ĐỊNH THỜI/BỘ ĐẾM

I. Mục đích

- Giao tiếp với màn hình LCD với Atmega128.
- Hiển thị văn bản lên màn hình LCD 16x2.

II. Phần cứng chủ yếu sử dụng trong bài thực hành

Trong bài thực hành này, chúng ta sử dụng các nút bấm Button của vi điều khiển, hiển thị tín hiệu với màn hình LCD, led RGB. Các phần cứng này có sẵn trên bảng mạch thực hành ở các vị trí 1,2,3 trên Hình 12.1.



Hình 12.1. Phần cứng sử dụng trong bài thực hành

III. Tóm tắt nội dung lý thuyết

Màn hình LCD đã trình bày ở bài 7.

Bộ định thời

Bộ định thời (Timer/Counter) trong vi điều khiển ATmega128 là các thanh ghi đếm xung nhịp (nội bộ hoặc ngoại vi), cho phép tạo độ trễ, đo thời gian, và tạo tín hiệu PWM, với ATmega128 có 2 Timer 8-bit (Timer0, Timer2) và 2 Timer 16-bit (Timer1, Timer3). Các timer này được cấu hình qua các thanh ghi điều khiển (TCCR), thanh ghi dữ liệu (TCNT), thanh ghi so sánh (OCR) để đặt giá trị, và thanh ghi ngắt (TIMSK) để bật cờ ngắt, giúp thực hiện các tác vụ hẹn giờ, đo xung, và tạo sóng (PWM) với nhiều chế độ hoạt động khác nhau (Normal, CTC, Fast PWM, Phase Correct PWM).

Các bộ định thời chính trên ATmega128:

Timer0 & Timer2: 8-bit, có 3 chế độ (Normal, CTC, PWM).

Timer1 & Timer3: 16-bit, mạnh mẽ hơn với 4 chế độ (Normal, CTC, Fast PWM, Phase Correct PWM) và hỗ trợ nhiều chức năng hơn (Capture, Input Capture).

Các thanh ghi quan trọng:

TCNT (Timer/Counter Register): Chứa giá trị đếm hiện tại.

TCCR (Timer/Counter Control Register): Cấu hình chế độ hoạt động (WGM), bộ chia tần số (CS), và ngắt (COM).

OCR (Output Compare Register): Giá trị so sánh để tạo ngắt hoặc PWM.

TIMSK (Timer/Counter Interrupt Mask Register): Cho phép bật các ngắt timer (TOIE, OCIE).

TIFR (Timer/Counter Interrupt Flag Register): Cờ ngắt (TOV, OCF) sẽ được đặt khi có sự kiện xảy ra (tràn bộ đếm, so sánh).

Chức năng cơ bản:

Tạo độ trễ: Cấu hình timer đếm đến một giá trị nhất định (thường dùng chế độ CTC) rồi tạo ngắt để delay.

Đo thời gian/xung: Dùng Timer 16-bit để đếm thời gian trôi qua.

Tạo tín hiệu PWM: Dùng các chế độ Fast PWM hoặc Phase Correct PWM để điều khiển độ sáng LED, tốc độ động cơ.

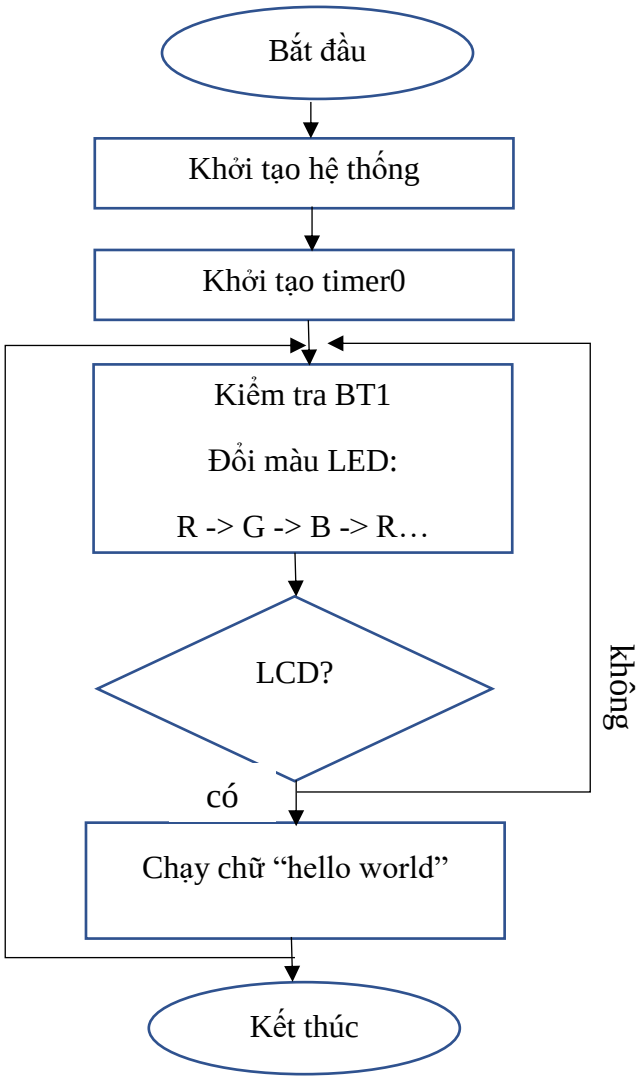
IV. Nội dung bài thực hành

Sử dụng LED 3 màu RGB, viết chương trình bấm BT.1 lần 1 đèn R sáng, lần 2 đèn R tắt đèn G sáng, lần 3 đèn G tắt đèn B sáng, lặp đi lặp lại liên tục. Trong lúc đang làm các tác vụ kể trên thì dòng chữ hello-world chạy từ trái sang phải trên màn hình (trong timer).

1. Sơ đồ kết nối

Tương tự như các bài trước.

2. Sơ đồ thuật toán



3. Mã chương trình

```
1   #include <io.h>
2   #include <alcd.h>
3   #include <delay.h>
4   #include <stdlib.h>
5   #define BT1 PINB.2
6   #define R_PIN PORTD4
7   #define G_PIN PORTD5
8   #define B_PIN PORTD6
9   #define LCD_BACKLIGHT_PIN PORTD7
10  const flash char message[] = "hello-world";
11  const unsigned char msg_len = 11;
12  const unsigned char max_col = 16;
13  unsigned char led_state = 3;
14  int scroll_position = 0;
15  volatile bit lcd_update_flag = 0;
16  unsigned char timer0_counter = 0;
17  // --- HAM KHOI TAO TIMER 0 ---
18  void timer0_init(void)
19  {
```

```
    // Clock/1024
    TCCR0 = 0x05;
    TCNT0 = 0x00;
    TIMSK |= (1<<TOIE0);
}
// --- DICH VU NGAT TRAN TIMER 0 ---
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
    timer0_counter++;
    if (timer0_counter >= 36) {
        timer0_counter = 0;
        scroll_position++;

        if (scroll_position > (max_col + msg_len)) {
            scroll_position = 0;
        }
        lcd_update_flag = 1;
    }
}
void check_button(void)
{
    if (BT1 == 0) {
        delay_ms(50);
        if (BT1 == 0) {
            led_state++;
            if (led_state > 2) {
                led_state = 0;
            }
            PORTD &= ~((1<<R_PIN) | (1<<G_PIN) | (1<<B_PIN));
            if (led_state == 0) {
                PORTD |= (1<<R_PIN); // RED
            } else if (led_state == 1) {
                PORTD |= (1<<G_PIN); // GREEN
            } else {
                PORTD |= (1<<B_PIN); // BLUE
            }
            while (BT1 == 0);
        }
    }
}
// --- HAM CHAY CHU TREN LCD ---
void scroll_text(void)
{
    int i;
    int col;
    lcd_gotoxy(0, 1);
    lcd_puts("               ");
    lcd_gotoxy(0, 0);
```

```
68        lcd_puts("LED State:      ");
69        lcd_gotoxy(11, 0);
70        if(led_state == 0) lcd_puts("RED ");
71        else if(led_state == 1) lcd_puts("GREEN");
72        else if(led_state == 2) lcd_puts("BLUE");
73        else lcd_puts("OFF "); // led_state = 3
74
75        for (i = 0; i < msg_len; i++) {
76            col = scroll_position - max_col + i;
77            if (col >= 0 && col < max_col) {
78                lcd_gotoxy(col, 1);
79                lcd_putchar(message[i]);
80            }
81        }
82        lcd_update_flag = 0;
83    }
84    void main(void)
85    {
86
87        DDRB &= ~(1<<DDB2);
88        PORTB |= (1<<PORTB2);
89
90        DDRD |= (1<<DDD4) | (1<<DDD5) | (1<<DDD6);
91
92        DDRD |= (1<<DDD7);
93
94        PORTD &= ~((1<<R_PIN) | (1<<G_PIN) | (1<<B_PIN));
95        PORTD |= (1<<LCD_BACKLIGHT_PIN);
96
97        lcd_init(16);
98        lcd_clear();
99        timer0_init();
100       #asm("sei")
101
102       while (1)
103       {
104           check_button();
105
106           if (lcd_update_flag) {
107               scroll_text();
108           }
109       }
110   }
```

4. Hướng dẫn thực hành

Trong đoạn chương trình trên, kết nối các chân theo hình 12.2. Để giao tiếp, chúng ta phải cấu hình trong chương trình CodeVision theo hình 12.4.

Sau khi soạn thảo chương trình xong, chúng ta biên dịch chương trình để tạo mã hex, tiếp tục sử dụng chương trình Progisp để burn mã hex của chương trình trên vào chíp và thao tác bấm BT1 quan sát trên màn hình LCD, led RGB.

BÀI 13. BỘ ĐỊNH THỜI/BỘ ĐẾM

I. Mục đích

- Giao tiếp với màn hình LCD với Atmega128.
- Hiển thị văn bản lên màn hình LCD 16x2.

II. Phần cứng chủ yếu sử dụng trong bài thực hành

Trong bài thực hành này, chúng ta sử dụng các nút bấm Button của vi điều khiển, hiển thị tín hiệu với màn hình LCD, led RGB.

III. Tóm tắt nội dung lý thuyết

Màn hình LCD đã trình bày ở bài 7.

Bộ định thời đã trình bày ở bài trước

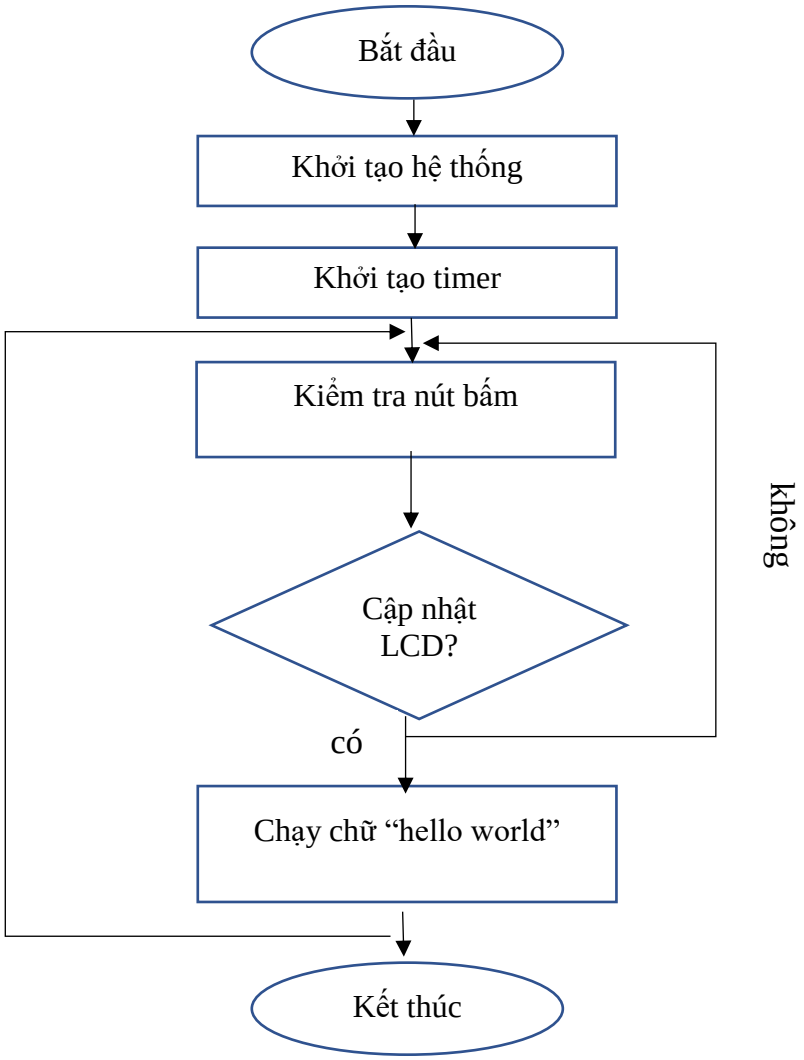
IV. Nội dung bài thực hành

Viết chương trình bấm phím BT-1 đèn sáng, BT 2 đèn tắt, BT 3 đèn LED nhấp nháy theo trình tự sáng - tối - sáng - tối - sáng mãi. Trong lúc đang làm các tác vụ kể trên thì dòng chữ hello-world chạy từ trái sang phải trên màn hình (trong timer).

1. Sơ đồ kết nối

Tương tự như các bài trước.

2. Sơ đồ thuật toán



3. Mã chương trình

```
1   #include <io.h>
2   #include <alcd.h>
3   #include <delay.h>
4
5   #define BT3 PINB.0  // Nhap nhay (Blink)
6   #define BT2 PINB.3  // Tat (Off)
7   #define BT1 PINB.2  // Sang (On)
8
9   #define LED_PIN PORTD4
10  #define LCD_BACKLIGHT_PIN PORTD7
11
12  // Che do hoat dong cua LED
13  #define MODE_OFF    0
14  #define MODE_ON     1
15  #define MODE_BLINK  2
16
17  const flash char message[] = "hello-world";
```

```
18  const unsigned char msg_len = 11;
19  const unsigned char max_col = 16;
20
21  // --- BIEN TOAN CUC ---
22  unsigned char led_mode = MODE_OFF;
23
24  int scroll_position = 0;
25  volatile bit lcd_update_flag = 0;
26
27  unsigned char timer0_scroll_counter = 0;
28  unsigned char timer0_blink_counter = 0;
29  volatile bit blink_toggle = 0;
30
31  // --- HAM KHOI TAO TIMER 0 ---
32  void timer0_init(void)
33  {
34     // Clock/1024
35     TCCR0 = 0x05;
36     TCNT0 = 0x00;
37     TIMSK |= (1<<TOIE0);
38  }
39  // --- NGAT TRAN TIMER 0 ---
40  interrupt [TIM0_OVF] void timer0_ovf_isr(void)
41  {
42     timer0_scroll_counter++;
43     if (timer0_scroll_counter >= 36) {
44        timer0_scroll_counter = 0;
45        scroll_position++;
46
47        if (scroll_position > (max_col + msg_len)) {
48           scroll_position = 0;
49        }
50        lcd_update_flag = 1;
51     }
52
53     if (led_mode == MODE_BLINK) {
54        timer0_blink_counter++;
55
56        if (timer0_blink_counter >= 15) {
57           timer0_blink_counter = 0;
58           blink_toggle = !blink_toggle;
59
60           if (blink_toggle) {
61              PORTD |= (1<<LED_PIN); // SÁNG
62           } else {
63              PORTD &= ~(1<<LED_PIN); // T?T
64           }
65        }
```

```
66      }
67  }
68
69  void check_buttons(void)
70  {
71      if (BT1 == 0) {
72          delay_ms(50);
73          if (BT1 == 0) {
74              led_mode = MODE_ON;
75              PORTD |= (1<<LED_PIN);
76              lcd_update_flag = 1;
77              while (BT1 == 0);
78          }
79      }
80
81      if (BT2 == 0) {
82          delay_ms(50);
83          if (BT2 == 0) {
84              led_mode = MODE_OFF;
85              PORTD &= ~(1<<LED_PIN);
86              lcd_update_flag = 1;
87              while (BT2 == 0);
88          }
89      }
90
91      if (BT3 == 0) {
92          delay_ms(50);
93          if (BT3 == 0) {
94              led_mode = MODE_BLINK;
95              timer0_blink_counter = 0;
96              blink_toggle = 0;
97              PORTD &= ~(1<<LED_PIN);
98              lcd_update_flag = 1;
99              while (BT3 == 0);
100         }
101     }
102 }
103
104 void scroll_text(void)
105 {
106     int i;
107     int col;
108
109     lcd_gotoxy(0, 0);
110     lcd_puts("LED Mode:       ");
111     lcd_gotoxy(10, 0);
112
113     if(led_mode == MODE_ON) lcd_puts("ON  ");
```

```
    else if(led_mode == MODE_OFF) lcd_puts("OFF ");
    else lcd_puts("BLINK");

    lcd_gotoxy(0, 1);
    lcd_puts("                ");

    for (i = 0; i < msg_len; i++) {
        col = scroll_position - max_col + i;
        if (col >= 0 && col < max_col) {
            lcd_gotoxy(col, 1);
            lcd_putchar(message[i]);
        }
    }

    lcd_update_flag = 0;
}
void main(void)
{

    DDRB &= ~((1<<DDB0) | (1<<DDB3) | (1<<DDB2));
    PORTB |= (1<<PORTB0) | (1<<PORTB3) | (1<<PORTB2);

    DDRD |= (1<<DDD4) | (1<<DDD7);
    PORTD &= ~(1<<LED_PIN);
    PORTD |= (1<<LCD_BACKLIGHT_PIN);

    lcd_init(16);
    lcd_clear();

    timer0_init();
    #asm("sei")
    scroll_text();
    while (1)
    {
        check_buttons();
        if (lcd_update_flag) {
            scroll_text();
        }
    }
```

4. Hướng dẫn thực hành

Trong đoạn chương trình trên, kết nối các chân theo hình 13.2. Để giao tiếp, chúng ta phải cấu hình trong chương trình CodeVision theo hình 13.4.

Sau khi soạn thảo chương trình xong, chúng ta biên dịch chương trình để tạo mã hex, tiếp tục sử dụng chương trình Progisp để burn mã hex của chương trình trên vào chíp và thao tác bấm BT1, 2, 3, quan sát trên màn hình LCD, led RGB.

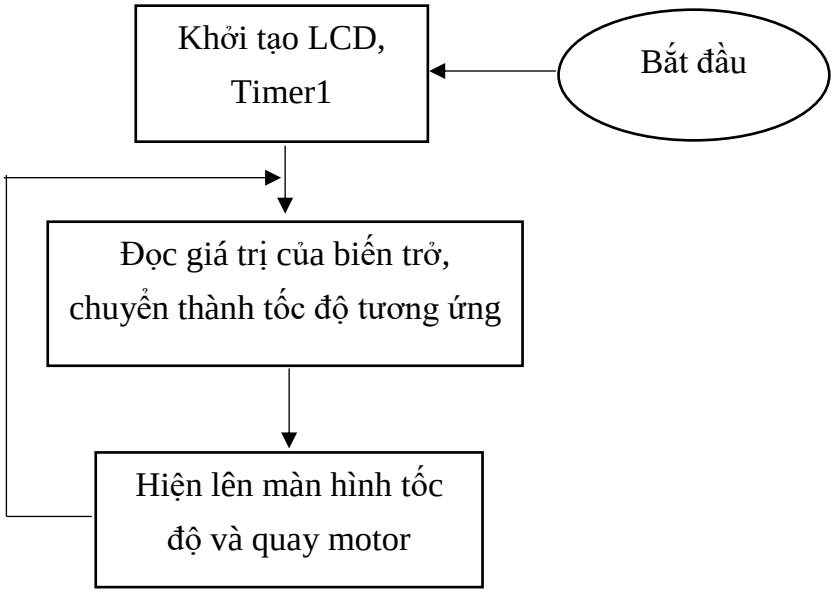
Bài 14

Nội dung thực hành: Viết chương trình mô phỏng tay ga của xe đạp điện/máy điện/oto (dẫn động 1 bánh -> điều khiển tốc độ động cơ theo thay đổi của chiết áp, hiển thị tốc độ tương đối lên GLCD/LCD (tùy chọn)

Gợi ý: sử dụng chiết áp thay đổi giá trị điện áp ADC vào một kênh nào đó và thay đổi giá trị PWM để điều khiển tốc độ động cơ)

Bài này tương tự bài 5.

1. Sơ đồ thuật toán



2. Mã chương trình

```
#include <io.h>
#include <delay.h>
#include <alcd.h>
#include <stdio.h>
#define BT1 PINB.2
#define BT2 PINB.3
#define BT3 PINB.0
int dir=0;
float speed=0;
unsigned int giatri;
char buffer[20];
void dkmotor(int chieu, int toc_do)
{
if(chieu==1)
{
delay_us(10);
ADCSRA|=1<<ADSC;
while((ADCSRA & 0b00010000)==0);
ADCSRA|=0x10;
return ADCW;
}
void main(void)
{
DDRD.7=1;
PORTD.7=1;
lcd_init(16);
DDRB=0b00110000;

PORTB=0b00101101;
ASSR=0x00;
TCCR0=0b01101011;
```

```
PORTB.5=1;
OCR0=255-255*toc_do/100;
}
if(chieu==0)
{

PORTB.5=0;
OCR0=255*toc_do/100;
}
}
unsigned int read_adc()
{
ADMUX=0x01;
ADMUX=0x01;
ADCSRA=(1<<ADEN|1<<ADPS1|1<<ADPS0);
while (1)
{
giatri=read_adc();
speed=(float) giatri*100/1023;
sprintf(buffer,"speed: %.0f%%",speed);
lcd_gotoxy(1,0);
lcd_puts(buffer);
lcd_clear();
dkmotor(dir,speed);
}
}
```

3. Hướng dẫn thực hành

Sau khi nạp chương trình, màn hình LCD hiện lên giá trị tốc độ quay của motor hiện tại. Khi thay đổi giá trị biến trở thì tốc độ quay của motor cũng thay đổi theo.

Bài 15

I. Mục đích

- Sử dụng bộ ADC của vi điều khiển để đọc quang trở và điều khiển động cơ

II. Phần cứng chủ yếu sử dụng trong bài thực hành

- Màn hình LCD
- Motor
- Quang trở

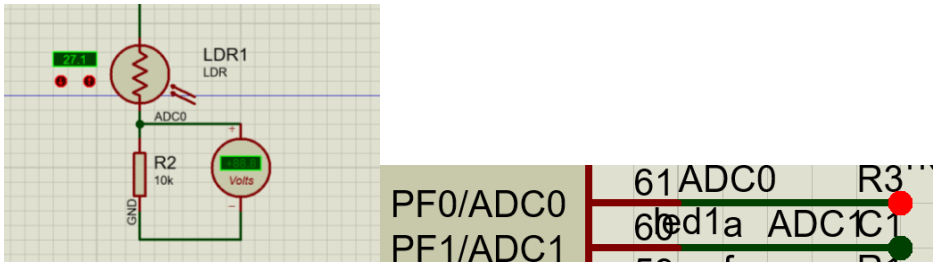
III. Tóm tắt lý thuyết

Quang trở (LDR - Light-Dependent Resistor) là một linh kiện điện tử có điện trở thay đổi theo cường độ ánh sáng: ánh sáng càng mạnh, điện trở càng giảm (dẫn điện tốt hơn), và ngược lại khi trời tối, điện trở tăng cao (khó dẫn điện). Nó hoạt động dựa trên nguyên lý quang dẫn và thường được dùng làm cảm biến ánh sáng trong các mạch tự động bật tắt đèn đường, báo động, đồng hồ, hoặc các thiết bị đo sáng nhờ cấu tạo đơn giản, giá rẻ và độ nhạy cao.

IV. Nội dung bài thực hành

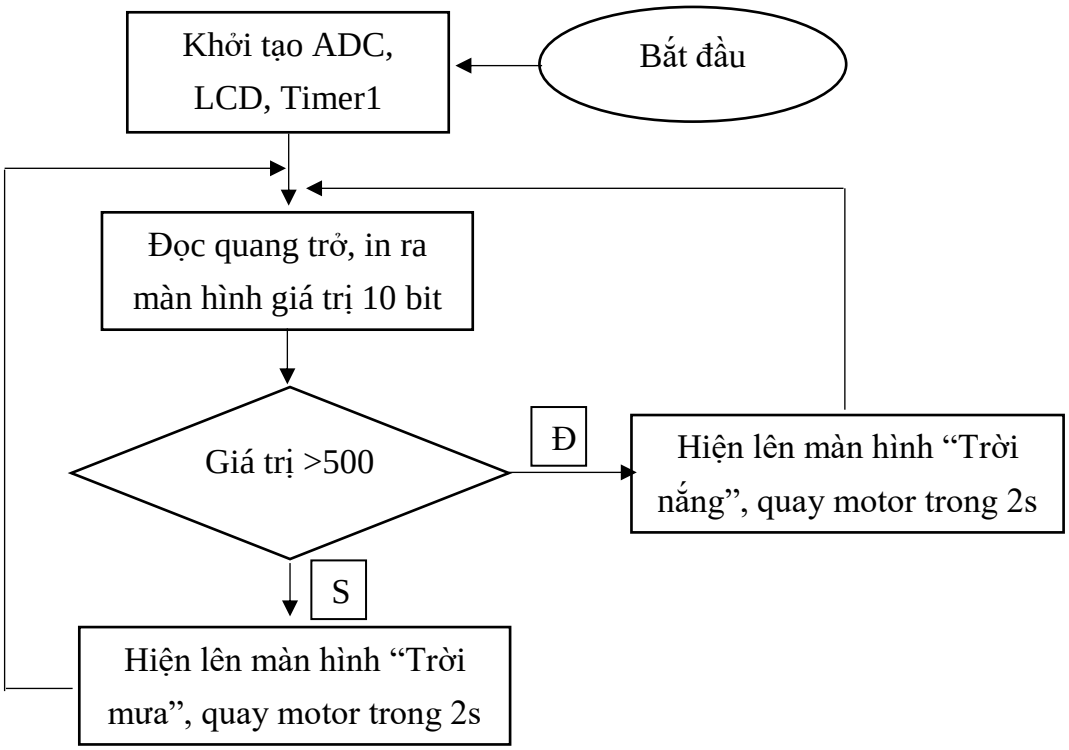
Viết chương trình mô phỏng giàn phơi khi trời tối thì cất quần áo, khi trời nắng thì đem quần áo ra phơi (điều khiển động cơ) (động cơ quay trong 1 hành trình nào đó, nếu quay nhiều quá -> đứt dây phơi) (hiển thị trạng thái trời nắng – tối lên GLCD/LCD) đọc ADC từ quang trở.

1. Sơ đồ kết nối



Hình 15.1. Sơ đồ kết nối chân ADC0 vào quang trở

2. Sơ đồ thuật toán



3. Mã chương trình

```
#include <io.h>
#include <delay.h>
#include <stdio.h>
#include <alcd.h>
#define BT3 PINB.0

unsigned int giatri,t=0;
unsigned char buffer[10];
unsigned int read_adc()
{
ADMUX=0x00;
delay_us(10);
ADCSRA|=1<<ADSC;
while((ADCSRA & 0b00010000)==0);
ADCSRA|=0x10;
return ADCW;
}
void dkmotor(int chieu, int toc_do)
{
if(chieu==1)
{
PORTB.4=1;
TCCR1B=(1<<WGM12|1<<CS11|1<<CS10);
DDRB=(1<<4|1<<5);
lcd_init(16);
ADMUX=0x00;
ADCSRA=(1<<ADEN|1<<ADPS1|1<<ADPS0);
while (1)
{
giatri=read_adc();
sprintf(buffer,"%d",giatri);
lcd_gotoxy(0,1);
lcd_puts(buffer);
delay_ms(100);
if (giatri<500&&t==1)

{
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("troi mua");

PORTC.3=1;
dkmotor(1,100);
delay_ms(2000);
t=0;
```

```
22  OCR1A=255-255*toc_do/100;
23  }
24  if(chieu==0)
25  {
26
27  PORTB.4=0;
28  OCR1A=255*toc_do/100;
29  }
30  }
31  void main(void)
32  {
33  DDRD.7=1;
34  DDRC.3=1;
35  PORTD.7=1;
36  TCCR1A=(1<<WGM10|1<<COM1A1);
58  dkmotor(1,0);
59  }
60  else if(giatri>500&&t==0)
61  {
62  lcd_clear();
63  lcd_gotoxy(0,0);
64  lcd_putsf("troi nang");
65  PORTC.3=0;
66  dkmotor(0,100);
67  delay_ms(2000);
68  t=1;
69  dkmotor(0,0);
70  }
71  }
72  }
```

4. Hướng dẫn thực hành

Thiết lập ADC đọc dữ liệu từ chân ADC0:

- Dòng 40: *ADMUX=0x00* nghĩa là ta đọc dữ liệu ở chân ADC0.

- Dòng 41: Bit ADEN được set lên 1 cho phép ADC hoạt động, 2 bit ADPS0 và ADPS1 được set lên 1 để chia tần số clock cho bộ ADC, tần số này là tần số lấy mẫu.

- Hàm *read_adc* (dòng 8 – 16): Ghi bit ADSC là 1 (dòng 12) để bắt đầu quá trình chuyển đổi. Khi 1 quá trình chuyển đổi hoàn tất thì bit này được ghi là 1. Như vậy ta đợi đến khi bit này được ghi là 0 thì ta sẽ đọc dữ liệu (dòng 13 -14). Dữ liệu trả về dưới dạng 2 thanh ghi ADCL và ADCH, kết hợp 2 thanh ghi này ta thu được giá trị ADC dưới dạng 10 bit.

Sau khi nạp chương trình, vi điều khiển sẽ liên tục đọc giá trị ADC của quang trở và hiện lên màn hình. Quy ước giá trị >500 là trời nắng, <500 là trời mưa. Khi lấy tay che, giá trị đọc được <500, màn hình hiện lên “trời mưa”, motor quay trong 2s. Khi bỏ tay, giá trị đọc được >500, màn hình hiện lên “trời nắng”, motor quay trong 2s.

BÀI 16

I. Mục đích

- Hiển thị thời gian thực lên LCD sử dụng DS1307

II. Phần cứng chủ yếu sử dụng trong bài

- Màn hình LCD

- Nút bấm

- Module thời gian thực DS1307

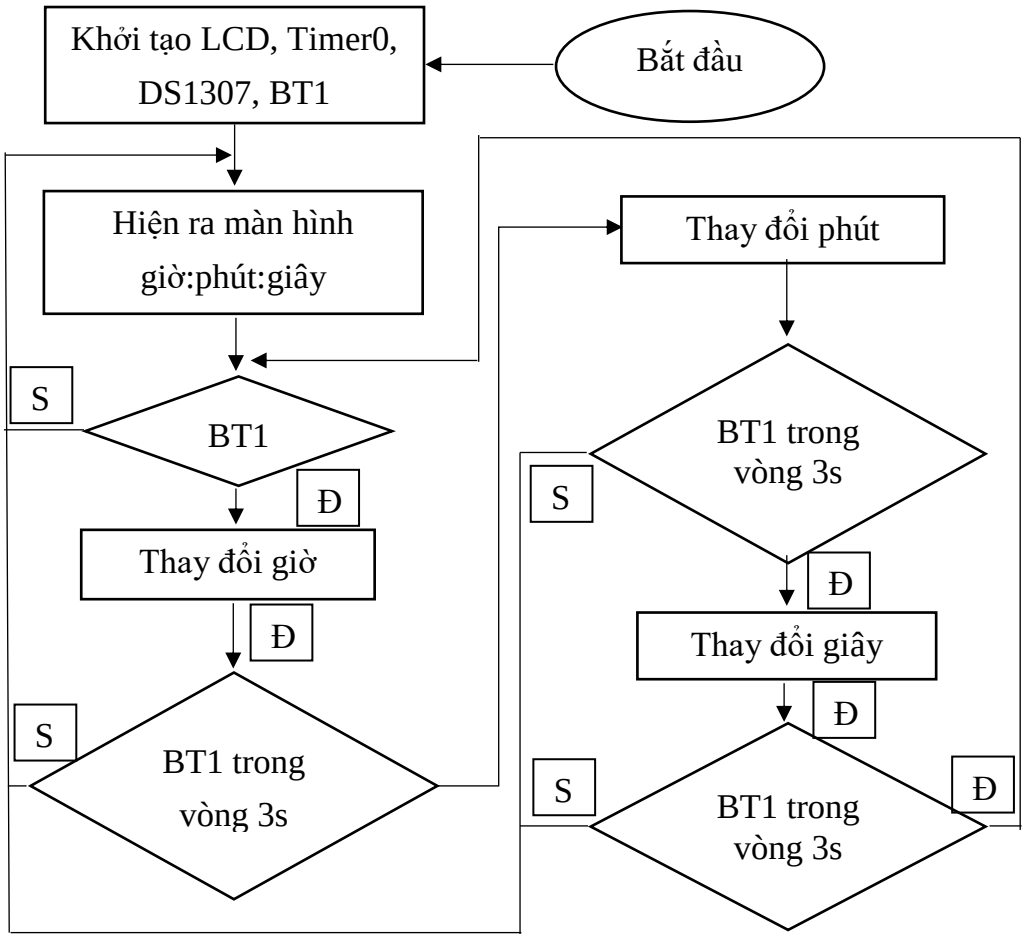
III. Tóm tắt lý thuyết

Phần module DS1307 và màn hình LCD đã được trình bày ở các bài trước.

IV. Nội dung bài thực hành

Viết chương trình sử dụng 1 nút bấm để điều chỉnh thời gian trên đồng hồ (nhớ lại cái đồng hồ casio đeo tay -> dùng duy nhất 1 nút bấm, khi bấm lần 1, ta chỉnh giờ, bấm lần 2 thì con trỏ nhảy sang đế chỉnh phút, bấm lần 3 Con trỏ chạy sang để chỉnh giây .... ngày, tháng, năm ...) Khi con trỏ đang ở vị trí chỉnh giờ, nếu ta bấm giữ button thì ngày/giờ/phút ... nó sẽ nhảy từ giá trị thấp nhất đến giá trị cao nhất (đối với năm chỉ thay đổi từ 2020 - 2030) làm sao để thoát. Khi bạn không tác động vào trong khoảng 3s thì chương trình sẽ nhảy ra và lưu giá trị cuối cùng các bạn bấm vào.

1. Sơ đồ thuật toán



2. Mã chương trình

```
1   #include <ds1307.h>
2   #include <alcd.h>
3   #include <stdio.h>
4   #include <delay.h>
5   #define BT1 PINB.2
6   int dem=0,x=0,y=0;
7   char display_buffer[17];
8   unsigned char
    hour,min,sec,hour1=12,min1=12,s
    ec1=12;
9   interrupt [TIM0_OVF] void
    timer0_ngat(void)
10  {
11  TCNT0=0x06;
12  dem=dem+1;
13  if (dem==3000)

14  {
46  if(dem>1000)
47  {
48  y++;
49  if(y==3) y=0;
50  }
51  }
52  if(x==1)
53  {

54  if(y==0)

55  {
56  if(BT1==0)   {hour1++; dem=0;}
57  if(hour1==24) hour1=0;
58  sprintf(display_buffer,"change
    hour:%02d",hour1);
59  lcd_clear();
```

```
15  x=0; y=0;
16  dem=0;
17  }
18  }
19  void main(void)
20  {
21  DDRD.7=1;
22  PORTD.7=1;
23  DDRB.2=0;
24  PORTB.2=1;
25  TCCR0=0x03;
26  TCNT0=0x06;
27  TIMSK=0x01;
28  #asm("sei")
29  lcd_init(16);
30  i2c_init();
31  rtc_init(3,1,0);
32  rtc_set_time(hour1,min1,sec1);
33  while (1)
34  {
35  if(x==0)
36  {
37  rtc_get_time(&hour,&min,&sec);
38  sprintf(display_buffer,"Time:%02d:
    %02d:%02d\n",hour,min,sec);
39  lcd_clear();
40  lcd_gotoxy(0,0);
41  lcd_puts(display_buffer);
42  }
43  if(BT1==0)
44  {
45  x=1;
60  lcd_gotoxy(0,0);
61  lcd_puts(display_buffer);
62  rtc_set_time(hour1,min1,sec1);
63  delay_ms(800);
64  }
65  if(y==1)
66  {
67  if(BT1==0)  {min1++; dem=0;}
68  if(min1==60)  min1=0;
69  sprintf(display_buffer,"change
    min:%02d",min1);
70  lcd_clear();
71  lcd_gotoxy(0,0);
72  lcd_puts(display_buffer);
73  rtc_set_time(hour1,min1,sec1);
74  delay_ms(800);
75  }
76  if(y==2)
77  {
78  if(BT1==0)  {sec1++; dem=0;}
79  if(sec1==60)  sec1=0;
80  sprintf(display_buffer,"change
    sec:%02d",sec1);
81  lcd_clear();
82  lcd_gotoxy(0,0);
83  lcd_puts(display_buffer);
84  rtc_set_time(hour1,min1,sec1);
85  delay_ms(800);
86  }
87  }
88  }
89  }
90
```

3. Hướng dẫn thực hành

Sau khi nạp chương trình, màn hình sẽ hiển thị giờ:phút:giây (dòng 35 – 42). Khi nhấn BT1, biến *x=1,* chuyển sang chế độ chỉnh thời gian, màn hình hiện lên "Change hour". Nhấn giữ BT1 (từ lúc đầu) thì giờ sẽ tăng lên trong phạm vi 0 – 23(dòng 54 – 64). Chờ 1 khoảng thời gian nhỏ 3s, nhấn giữ BT1 lần nữa thì màn hình hiện lên "Change minute" và phút sẽ tăng lên trong phạm vi 0 – 59 (dòng 65 – 75). Nhấn giữ lần nữa thì giây sẽ thay đổi (dòn 76 – 86). Nhẫn giữ lần nữa thì quay lại chỉnh giờ. Trong vòng 3s (đếm bởi hàm ngắt tràn của Timer0) nếu không bấm nút, biến *x=0,* quay trở về chế độ hiện thị giờ:phút:giây với thời gian được cập nhật từ các lần chỉnh giờ trước đó.