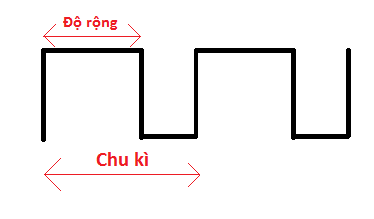
**[Matrix LED] Bài 9: Kĩ thuật PWM – hiệu ứng led sao băng**

[6 Tháng Chín, 2019](http://iot47.com/2019/09/) [Đào Nguyện](http://iot47.com/author/daonguyen/) [Ma trận LED](http://iot47.com/category/ma-tran-led/) [1](http://iot47.com/matrix-led-bai-9-ki-thuat-pwm-hieu-ung-led-sao-bang/#mh-comments)

xung PWM

Chúng ta đã quen với việc điều khiển led bằng các tín hiệu bật tắt, còn để điểu khiển độ sáng của LED thì sao ? Muốn LED thay đổi độ sáng thì phải thay đổi điện áp cấp cho LED, nhưng vi điều khiển thì chỉ có thể cấp ra tín hiệu 1 và 0 tương ứng 5V hoặc 0V. Có 1 kĩ thuật để thay đổi điện áp cấp ra cho LED đó là sử dụng PWM  
Bản chất, PWM là liên tục đưa ra các mức logic 1 và 0 ở 1 tốc độ rất cao. Khoảng thời gian chênh lệch giữa thời gian tồn tại mức 1 và 0 quyết định điện áp cấp, cái này chính là điều chế độ rộng xung PWM.  
Ví dụ, 1 chương trình đơn giản để tạo xung PWM ra 1 chân của vi điều khiển như sau, mình dẽ demo với 1 con LED đơn thôi đã nhé !



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | while(1)  {    P0\_0 = 1;    delay\_ms(1);    P0\_0 = 0;    delay\_ms(9);  } |

Chương trình trên liên tục bật tắt chân P0\_0 với thời gian tồn tại mức 1 là 1ms, thời gian tồn tại mức 0 là 9ms. Do vậy xung này có chu kì 10ms <=> 100Hz. Mức cao chiếm 10% của chu kì nên LED sẽ chỉ sáng với 10% độ sáng tối đa của nó ! Để thay đổi độ sáng, ta chỉ cần điều chế lại độ rộng này, nói đơn giản là thay đổi thời gian delay(1). Tuy nhiên cái delay bên trên tăng thêm bao nhiều thì cái delay phía dưới giảm đi bây nhiêu để đảm bảo tổng 2 cái delay là 10ms nhé. Điều này sẽ đảm bảo tần số PWM của ta luôn ổn định.

Ví dụ chương trình làm đèn sáng lên dần rồi tối đi dần



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | int i,y;  while(1)  {    //làm led sáng dần lên    for(i=0;i<=10;i++)  //vòng lặp for này sẽ làm thay đổi độ rộng xung    {       for(y=0;y<100;y++) //vòng lặp for này tạo ra 1 độ trễ nhất định, chính là tốc độ mà đèn sáng dần lên đó       {          P0\_0 = 1;          delay\_ms(i);          P0\_0 = 0;          delay\_ms(10-i);       }    }    //làm led tối dần đi    for(i=10;i>=0;i--)  //vòng lặp for này sẽ làm thay đổi độ rộng xung    {       for(y=0;y<100;y++) //vòng lặp for này tạo ra 1 độ trễ nhất định, chính là tốc độ mà đèn tối dần đó       {          P0\_0 = 1;          delay\_ms(i);          P0\_0 = 0;          delay\_ms(10-i);       }    }  } |

Kĩ thuật trên gọi là PWM mềm. Thực chất hầu hết các bộ vi điều khiển đều trang bị các bộ PWM cứng, nó là 1 phần cứng độc lập với MCU, chúng ta chỉ cần cấu hình các thông số tần số và độ rộng cho nó là co ngay xung PWM, tuy nhiện nó chỉ gjới hạn 1 vài kênh PWM thôi. Nói chung, PWM cứng thường dùng với các mục đích như điều khiển động cơ. Còn điều khiển LED thì số lượng led của chúng ta rất nhiều, không 1 vi điều khiển nào có nhiều bộ PWM cứng như thế cả. Do vậy, để điều khiển LED bằng PWM, chúng ta sẽ sử dụng PWM mềm nhé !

PWM cần tuân thủ chính xác về tần số và độ rộng, nếu lập trình như code bên trên, chúng ta sẽ chỉ điều khiển được 1 kênh LED và không thể làm được công việc khác. Do vậy mình sẽ sử dụng ngắt timer cho bộ PWM mềm

**PWM mềm bằng phương pháp ngắt timer**

Đến lúc này, 1 thông số chúng ta cần đặc biệt quan tâm tới PWM là độ phân giải của xung. Trở lại code PWM mềm mình đã chia sẻ bên trên, các bạn có thể thấy độ sáng của LED được điều khiển với vòng lặp for chạy từ 0 tới 10. Có tổng cộng 11 giá trị sáng trong dải số này. Do vậy số mức sáng khác nhau mà con LED chúng ta vừa điều khiển là 11. Độ phân giải là 1 thông số rất quan trọng khi điều khiển LED. Độ phân giải càng cao đương nhiên càng tốt nhưng sẽ càng hao phí tài nguyên của vi điều khiển, hãy thử code để kiểm nghiệm nhé !

Mình sẽ sử dụng bộ timer 1 của vi điều khiển 89s52 để demo 1 chương trình tạo PWM trên chân P0\_0 với mức sáng là 11 mức



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | int mucsangmax = 10; //mức sáng tối đa mà ta muốn  int dem;  // 1 biến con chạy  int dosangLED = 5; // LED sẽ sáng ở mức số 5 ( khoảng 50%)  void T1\_ISR() interrupt 3 // Dung timer 1 để tạo PWM ( 1ms tự gọi 1 lần)  {  TH1=0xFC; // Nap gia tri cho TH1  TL1=0x17; // Nap gia tri cho TL1  (đếm 1000 xung sẽ tràn, thạch anh 12Mhz)  if(dosangLED &gt; dem) P0\_0 = 1; //xuất mức 1 nếu con chạy vẫn nhỏ hơn mức sáng cài đặt  else  P0\_0 = 0;  dem++;if(dem&gt;mucsangmax)dem=0;  } |

Đoạn chương trình này tạo ra 1 PWM trên P0\_0, tuy nhiên nó được tự động thực hiện nhờ vào ngắt timer chứ không cần bỏ vào hàm main như chương trình đã viết bên trên nữa. Do đó, ta có thể thoải mái làm cộng việc khác trong main mà không lo ảnh hướng tới tần số của xung. Mình sẽ giải thích code trên 1 chút

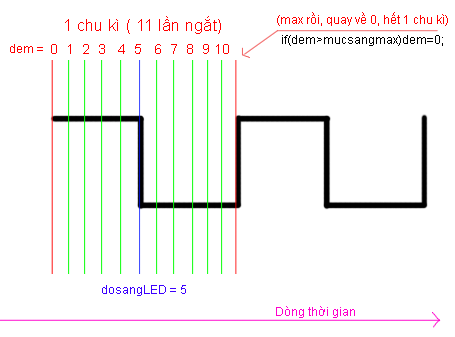
Do giá trị đếm nạp vào thanh ghi TH và TL là 0xFC17 = 64 535 nên bộ timer sẽ đếm 1000 xung rồi tràn. Do thạch anh là 12Mhz nên 1 xung đếm sẽ mất 1us. 1000 xung sẽ mất 1mili giây (1ms). Nên hàm ngắt này sẽ tự gọi 1ms 1 lần

Biến mucsangmax chính là độ phẩn giải mà ta mong muốn. Do mình cần 11 mức sáng khác nhau nên mình để mucsangmax = 10. Tại sao là bị trừ đi ?. Bởi biến đếm của chúng ta bắt đầu từ 0

Biến dosangLED là độ sáng mà chúng ta mong muốn trên LED và nó nằm trong khoảng từ 0 -> mucsangmax

Công việc bây giờ là xuất các giá trị 1 và 0 ra GPIO. Bằng việc so sánh xem giá trị sáng của LED có lơn biến dem không, nếu giá trị sáng vẫn lớn hơn thì out ra 1. Ngược lại thì out ra 0. Biến dem chạy tới mucsangmax là kết thúc 1 chu kì quét rồi đó, nên sẽ quay lại làm lại từ đầu ( dem=0 )

Các bạn có thể nhìn hình ảnh minh họa cho dễ hiểu

OK. bây giờ trong hàm main ta chỉ cần thay đổi dosangLED là được

Vi dụ chương trình đầy đủ sáng dần tắt dần 1 LED trên chân P0\_0 dùng ngắt timer



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | #include <REGX52.H>    #define led1 P2\_0    void delay\_ms(unsigned int t) //hàm delay     {        unsigned int x,y;        for(x=0;x<t;x++)          {              for(y=0;y<123;y++);          }     }      int mucsangmax = 10; //mức sáng tối đa mà ta muốn  int dem;  // 1 biến con chạy  int dosangLED = 5; // LED sáng sáng mức số 5 ( khỏang 50%)  void T1\_ISR() interrupt 3 // Dung timer 1 ngắt tạo pwm)  {  TH1=0xFC; // Nap gia tri cho TH1  TL1=0x17; // Nap gia tri cho TL1  (d?m 1000 xung s? tràn, th?ch anh 12Mhz)  if(dosangLED > dem) led1 = 1; //xuất mức 1 nếu độ sáng lớn hơn biến con chạy  else  led1 = 0;  //ngược lại xuất mức 0  dem++;if(dem>mucsangmax)dem=0;  //nếu dem vượt quá độ sáng tối đa thì hết 1 chu kì quét, reset lại  }    void main()  {  int i;  TMOD=0x10;     // khoi tao ngat T1, 16bit  ET1=1;     // cho phep ngat T1  TF1=0;     // xoa co ngat T1  TR1=1;     // khoi dong T1  EA = 1;        // cho phep ngat toan cuc  while(1)  {    for(i=0;i<=10;i++) //vòng lặp sáng dần    {     dosangLED=i;     delay\_ms(100);    }    for(i=10;i>=0;i--) //vòng lặp tối dần    {     dosangLED=i;     delay\_ms(100);    }  }  } |

Đó là 11 mức sáng, vậy muốn 100 mức sáng khác nhau thì sao ? Đơn giản chỉ cần tăng biến dosangmax lên 99 là xong :D, lúc này biến dem sẽ chạy tới 100 mới kết thúc 1 chu kì, do vậy 1 chu kì sẽ có tới 100 lần ngắt lận, tương tự cho 1000 mức sáng. Khi tăng số mức sáng sẽ kéo theo số lần ngắt trong 1 giây tăng lên, hãy nhớ rằng các lệnh con con tròng hàm ngắt cũng chiếm 1 chút thời gian xử lí và nếu ngắt tới 1000 lần cho 1 chu kì thì thời gian thực hiện các lệnh đó sẽ đáng kể hẳn lên. Dẵn tới việc vi điều khiển sẽ hao tốn thời gian cho các lệnh này làm tốc độ thực hiện chương trình ở hàm main bị chậm lại đáng kể.

**Hiệu ứng LED sao băng**

Hiệu ứng LED sao băng là 1 trong những hiệu ứng đẹp và độc đáo ! Với pwm mềm bằng timer, chúng ta sẽ dễ dàng tạo ra hiệu ứng này. Mình sẽ tạo 1 sao băng rượt đuổi trên 8 con LED. Do vậy cần 8 biến lưu trữ độ sáng của chúng, nhưng thay vì tạo 8 biến thì mình sẽ tạo 1 mảng dữ liệu gồm 8 phần tử cho dễ thao tác, và mình sẽ gán sẵn 8 mức sáng khác nhau vào mảng



|  |  |
| --- | --- |
| 1 | unsigned char led[8]={1,3,6,9,13,17,25,30}; //do sang cua 8 LED |

Bây giờ định nghĩa các chân LED tương ứng



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | #define led1 P2\_0  #define led2 P2\_1  #define led3 P2\_2  #define led4 P2\_3  #define led5 P2\_4  #define led6 P2\_5  #define led7 P2\_6  #define led8 P2\_7 |

Biến độ sáng max mình sẽ tăng lênh thành 30 để có nhiều mức sáng khác nhau hơn, do đó chu kì lúc này sẽ dài hơn -> tần số quét giảm đi. Để tăng tần số lên mình sẽ nạp vào 2 thanh ghi TH1 TL1 giá trị là 0xFE0B (0.5ms ngắt 1 lần – nhanh gấp đổi code bên trên)

OK. Giờ chỉnh lại chương trình ngắt thành như sau



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | void T1\_ISR() interrupt 3 // Dung timer 1 tao PWM 0.5ms goi 1 lan)  {  if(led[0] > dem) led1 = bat; //xuat du lieu ra chan LED 1  else  led1 = tat;  if(led[1] > dem) led2 = bat; //xuat du lieu ra chan LED 2  else  led2 = tat;  if(led[2] > dem) led3 = bat; //xuat du lieu ra chan LED 3  else  led3 = tat;  if(led[3] > dem) led4 = bat; //xuat du lieu ra chan LED 4  else  led4 = tat;  if(led[4] > dem) led5 = bat; //xuat du lieu ra chan LED 5  else  led5 = tat;  if(led[5] > dem) led6 = bat; //xuat du lieu ra chan LED 6  else  led6 = tat;  if(led[6] > dem) led7 = bat; //xuat du lieu ra chan LED 7  else  led7 = tat;  if(led[7] > dem) led8 = bat; //xuat du lieu ra chan LED 8  else  led8 = tat;  dem++;if(dem>mucsangmax)dem=0;  TH1=0xFE; // Nap gia tri cho TH1  TL1=0x0B; // Nap gia tri cho TL1  (dem 500 xung tràn, thach anh 12Mhz)  } |

Chương trình ngắt trên giống như ngắt với 1 con LED. Chỉ là có nhiều LED hơn nên mình sẽ phải kiểm tra và xuất tín hiệu ra ra chân nhiều hơn thôi !

Toàn bộ CODE như sau:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64 | #include <REGX52.H>    #define led1 P2\_7  #define led2 P2\_6  #define led3 P2\_5  #define led4 P2\_4  #define led5 P2\_3  #define led6 P2\_2  #define led7 P2\_1  #define led8 P2\_0    #define bat 0  #define tat 1    void delay\_ms(unsigned int t) //hàm delay     {        unsigned int x,y;        for(x=0;x<t;x++)          {              for(y=0;y<123;y++);          }     }      int mucsangmax = 30; //muc sáng toi da mà ta muon = 30  int dem;  // 1 bien con chay  unsigned char led[8]={1,3,6,9,13,17,25,30}; //do sang cua 8 LED    void T1\_ISR() interrupt 3 // Dung timer 1 d? t?o PWM 1ms g?i 1 l?n)  {  if(led[0] > dem) led1 = bat; //xuat du lieu ra chan LED 1  else  led1 = tat;  if(led[1] > dem) led2 = bat; //xuat du lieu ra chan LED 2  else  led2 = tat;  if(led[2] > dem) led3 = bat; //xuat du lieu ra chan LED 3  else  led3 = tat;  if(led[3] > dem) led4 = bat; //xuat du lieu ra chan LED 4  else  led4 = tat;  if(led[4] > dem) led5 = bat; //xuat du lieu ra chan LED 5  else  led5 = tat;  if(led[5] > dem) led6 = bat; //xuat du lieu ra chan LED 6  else  led6 = tat;  if(led[6] > dem) led7 = bat; //xuat du lieu ra chan LED 7  else  led7 = tat;  if(led[7] > dem) led8 = bat; //xuat du lieu ra chan LED 8  else  led8 = tat;  dem++;if(dem>mucsangmax)dem=0;  TH1=0xFE; // Nap gia tri cho TH1  TL1=0x0B; // Nap gia tri cho TL1  (dem 500 xung tràn, thach anh 12Mhz)  }    void main()  {  int i;  TMOD=0x10;     // khoi tao ngat T1, 16bit  ET1=1;     // cho phep ngat T1  TF1=0;     // xoa co ngat T1  TR1=1;     // khoi dong T1  EA = 1;        // cho phep ngat toan cuc  while(1)  {    }  } |

Kết quả 8 led sáng tướng ứng với mức sáng đà đặt trong mảng dữ liệu

Giờ mình sẽ code hàm main cho 8 led này chạy rượt đuổi để tạo ra hiệu ứng sao băng. Mình sẽ tạo thêm 1 mảng là hằng số để lưu giữ liệu sáng của led.



|  |  |
| --- | --- |
| 1 | const unsigned char data\_led[8]={1,3,6,9,13,17,25,30}; //do sang luu tru |

OK. Trong hàm main mình sẽ viết vòng lặp để lấy 8 độ sáng đã lưu trong mản data\_led ghi vào mảng led

C



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | for(k=7;k>-12;k--) //-12 la khoang cach giua cac tia sao bang, cang lon khoang cach cang xa  {      for(i=7;i>0;i--)led[i]=led[i-1]; //dich array led sang phai >> 1 lan    if(k>=0)led[0] = data\_led[k]; // lay do sang trong data\_led ghi vao led[0]    else led[0]=0; //tat led 0    delay\_ms(50);  } |

Demo hiệu ứng led sao băng

Các bạn thử thay đổi biến -12 để thấy tốc độ xuất hiện của sao băng nhé. hàm delay chính là tốc độ chạy của sao băng

Vòng lặp ngoài cùng bắt đâu từ 7 tương ứng thao tác lên con led ngoài cùng đầu tiên ( led[7] ). Tiếp đến vòng lặp con 7 lần mình sẽ dịch phải toàn bộ mảng led ( tức là con led bên phải sẽ lấy độ sáng của con led bên trái) Cứ thế tới con led[1] sẽ lấy độ sáng của con led[0]. Tới led[0] do không con con led nào bên trái nó nữa nên nó sẽ chạy vào dữ liệu được lưu trong data\_led để lấy, tương ứng với vị trí của biến k . nếu k giảm xuống âm thì sẽ cho led[0] = 0 luôn !

Bạn càng giảm k xuống nhỏ thì thời gian con led[0] bị tắt càng lâu hơn ! Dẫn tơi sao băng xuất hiện chậm hơn

FULL CODE đây nhé

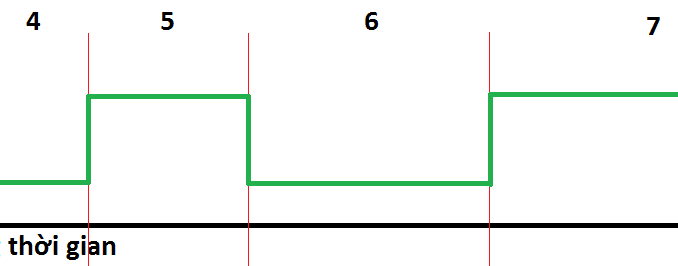
C



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | #include <REGX52.H>    #define led1 P2\_7  #define led2 P2\_6  #define led3 P2\_5  #define led4 P2\_4  #define led5 P2\_3  #define led6 P2\_2  #define led7 P2\_1  #define led8 P2\_0    #define bat 0  #define tat 1    void delay\_ms(unsigned int t) //hàm delay     {        unsigned int x,y;        for(x=0;x<t;x++)          {              for(y=0;y<123;y++);          }     }      int mucsangmax = 30; //muc sáng toi da mà ta muon = 30  int dem;  // 1 bien con chay  const unsigned char data\_led[8]={1,3,6,9,20,30}; //do sang cua 8 LED  unsigned char led[8]; //do sang cua 8 LED    void T1\_ISR() interrupt 3 // Dung timer 1 d? t?o PWM 1ms g?i 1 l?n)  {  if(led[0] > dem) led1 = bat; //xuat du lieu ra chan LED 1  else  led1 = tat;  if(led[1] > dem) led2 = bat; //xuat du lieu ra chan LED 2  else  led2 = tat;  if(led[2] > dem) led3 = bat; //xuat du lieu ra chan LED 3  else  led3 = tat;  if(led[3] > dem) led4 = bat; //xuat du lieu ra chan LED 4  else  led4 = tat;  if(led[4] > dem) led5 = bat; //xuat du lieu ra chan LED 5  else  led5 = tat;  if(led[5] > dem) led6 = bat; //xuat du lieu ra chan LED 6  else  led6 = tat;  if(led[6] > dem) led7 = bat; //xuat du lieu ra chan LED 7  else  led7 = tat;  if(led[7] > dem) led8 = bat; //xuat du lieu ra chan LED 8  else  led8 = tat;  dem++;if(dem>mucsangmax)dem=0;  TH1=0xFE; // Nap gia tri cho TH1  TL1=0x0B; // Nap gia tri cho TL1  (dem 500 xung tràn, thach anh 12Mhz)  }    void main()  {  signed int i,k;  TMOD=0x10;     // khoi tao ngat T1, 16bit  ET1=1;     // cho phep ngat T1  TF1=0;     // xoa co ngat T1  TR1=1;     // khoi dong T1  EA = 1;        // cho phep ngat toan cuc  while(1)  {  for(k=7;k>-12;k--) //-12 la khoang cach giua cac tia sao bang, cang lon khoang cach cang xa  {      for(i=7;i>0;i--)led[i]=led[i-1]; //dich array led sang phai >> 1 lan    if(k>=0)led[0] = data\_led[k]; // lay do sang trong data\_led ghi vao led[0]    else led[0]=0; //tat led 0    delay\_ms(50);  }  }  } |

# [Matrix LED] Bài 10: Kĩ thuật PWM – Điều chế động rộng bằng phương pháp B.A.M

[18 Tháng Mười Hai, 2019](http://iot47.com/2019/12/) [Đào Nguyện](http://iot47.com/author/daonguyen/) [Ma trận LED](http://iot47.com/category/ma-tran-led/) [0](http://iot47.com/matrix-led-bai-10-ki-thuat-pwm-dieu-che-dong-rong-bang-phuong-phap-b-a-m/#mh-comments)

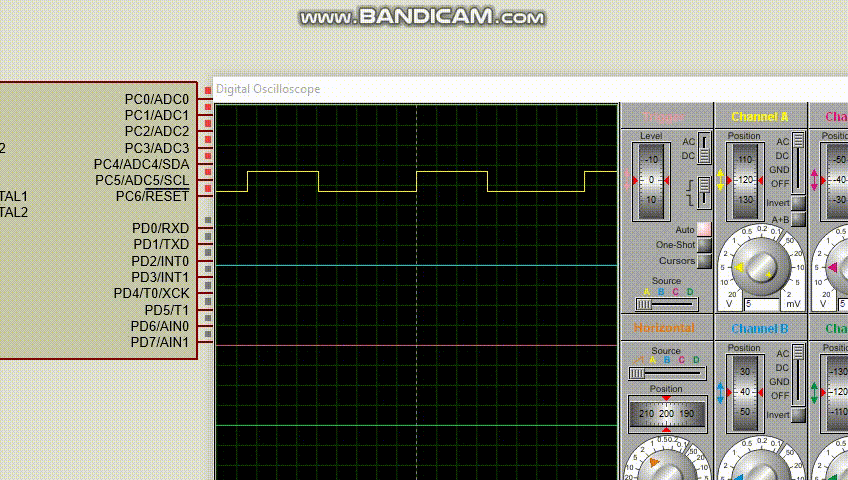


Trong [**phần trước**](http://iot47.com/matrix-led-bai-9-ki-thuat-pwm-hieu-ung-led-sao-bang/) mình đã demo giới thiệu của phương pháp PWM mềm, thuật toán ở bài trước đó khá ổn nếu điều khiển PWM ra 1 số lượng GPIO hữu hạn ! Mình sẽ thử lấy 1 ví dụ với PWM ra LED 256 mức sáng bằng vi điều khiển ATmega8, chương trình tạo PWM thông thường sẽ như sau:

C



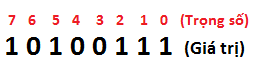
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | #include <mega8.h>  #include <delay.h>  unsigned char dem;  unsigned char sang=200;  interrupt [TIM1\_OVF] void ngattimer1\_1ms(void)  {    if(sang>=dem)PORTC.0=1;    else PORTC.0=0;    dem++;    TCNT1H=0xFE;      // Khoi tao lai gia tri cho TCNT1    TCNT1L=0x0B;  }    void main(void)  {    PORTC=0xff;  DDRC=0xff;    // KHOI TAO TIMER1  TCCR1A=0x00;  TCCR1B=0x01;        // Chia tan so cho 1    TCNT1H=0xFE;      // Khoi tao lai gia tri cho TCNT1  TCNT1L=0x0B;    TIMSK=0x04;    // Kich hoat ngat tran TIMER1  #asm("sei")    // Kich hoat ngat toan cuc    while (1)        {          if(sang<255)sang++;          else sang=0;          delay\_ms(50);        }  } |

Dạng sóng của phương pháp PWM thông thường

Phương pháp [**PWM mềm thông thường**](http://iot47.com/matrix-led-bai-9-ki-thuat-pwm-hieu-ung-led-sao-bang/) này sẽ rất hao tổn tài nguyên của MCU nếu bạn điều khiển quá nhiều GPIO và số lượng bit PWM lớn do mỗi mức PWM là 1 lần ngắt, nói cách khác, muốn tạo ra 256 mức phải ngắt 256 lần. Có 1 thủ thuật khác hay hơn để tạo ra 256 mức PWM (8bit PWM) chỉ với 8 lần ngắt nhờ lợi dụng đặc điểm của số nhị phân.

Ví dụ, mình cần tạo ra 1 PWM có max 256 mức, độ rộng xung ở mức cao là 167, cách tạo ra PWM bằng phương pháp B.A.M sẽ như sau:

167(thập phân)=10100111(nhị phân)  
Trọng số của các bit nhị phân được đánh từ trái sang phải bắt đầu từ 0



Lần ngắt thứ 1: Bit trọng số 0 có giá trị là 1 => đưa chân GPIO lên logic 1 > delay 1 tick  
Lần ngắt thứ 2: Bit trọng số 1 có giá trị là 1 => đưa chân GPIO lên logic 1 > delay 2 tick  
Lần ngắt thứ 3: Bit trọng số 2 có giá trị là 1 => đưa chân GPIO lên logic 1 > delay 4 tick  
Lần ngắt thứ 4: Bit trọng số 3 có giá trị là 0 => đưa chân GPIO lên logic 0 > delay 8 tick  
Lần ngắt thứ 5: Bit trọng số 4 có giá trị là 0 => đưa chân GPIO lên logic 0 > delay 16 tick  
Lần ngắt thứ 6: Bit trọng số 5 có giá trị là 1 => đưa chân GPIO lên logic 1 > delay 32 tick  
Lần ngắt thứ 7: Bit trọng số 6 có giá trị là 0 => đưa chân GPIO lên logic 0 > delay 64 tick  
Lần ngắt thứ 8: Bit trọng số 7 có giá trị là 1 => đưa chân GPIO lên logic 1 > delay 128 tick

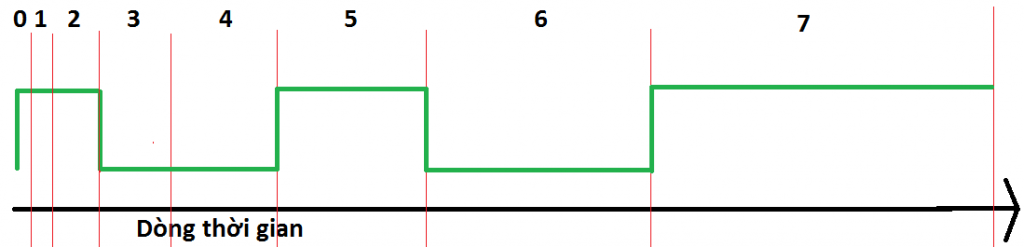
Như vậy là kết thúc 1 chu kì PWM chỉ với 8 lần ngắt. Mình sẽ demo chỉnh sửa code trên sang thuật toán B.A.M



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | #include <mega8.h>  #include <delay.h>    unsigned char vitri\_bit=0;  unsigned char sang=167;  unsigned int vl\_bit\_timer[8]={0xFFB1,0xFF63,0xFEC7,0xFD8E,0xFB1D,0xF63B,0xEC77,0xD8EF};  //unsigned int vl\_bit\_timer[8]={0xFEC7,0xFD8F,0xFB1F,0xF63F,0xEC77,0xD8FF,0xB1DF,0x64BF};  //8 lan  interrupt [TIM1\_OVF] void ngattimer1\_1ms(void)  {    if((sang & (1<<vitri\_bit)))PORTC.0=1;    else PORTC.0=0;      TCNT1H=vl\_bit\_timer[vitri\_bit]>>8;      // Khoi tao lai gia tri cho TCNT1     ( bit 0);    TCNT1L=vl\_bit\_timer[vitri\_bit];      vitri\_bit++;if(vitri\_bit==7)vitri\_bit=0;  //tang bit len 1  }    void main(void)  {  PORTC=0xff;  DDRC=0xff;    // KHOI TAO TIMER1  TCCR1A=0x00;  TCCR1B=0x01;        // Chia tan so cho 1    TIMSK=0x04;    // Kich hoat ngat tran TIMER1    #asm("sei")    // Kich hoat ngat toan cuc    while (1)        {          }  } |

Ở chương trình trên, 1 biến vitri\_bit được tạo để giám sát xem chúng ta đang cần làm việc với bit nào, mỗi lần ngắt mình sẽ nạp lại giá trị cho timer để thay đổi thời gian nhảy vào lần ngắt tiếp theo (tạo delay)

Dạng sóng của phương pháp B.A.M sẽ như sau:



Tổng thời gian ở mức cao sẽ tương đường với phương pháp PWM thông thường