*< Tác giả PhùngThái Sơn, Hà NAM, Việt Nam. Trang web: arduino.vn/NickChung>*

*<Mình xin giữ bản quyền tài liệu này, bạn có thể sử dụng tài cho mọi mục đích nhưng cần trích thông tin tác giả, cảm ơn bạn đã đón đọc!>*

*https://github.com/NickChungVietNam/ALL\_EEPROM\_24Cxx\_ATMEL\_ARDUINO\_AT\_MASTER*

Hướng dẫn sử dụng ic eeprom 24Cxx của Atmel.

Lời dẫn:

Bấy lâu nay bạn đã sử dụng sai eeprom của arduino?

EEPROM là còn được hiểu như một thẻ nhớ có thể lưu trữ dữ liệu ngay cả khi mất điện.

Bộ nhớ EEPROM có sẵn của arduino có số lần ghi/xóa 100000 lần, nếu dùng hết số lần ghi cho phép, eeprom sẽ hỏng. Để khắc phục điều này, chúng ta cần sử dụng một ic eeprom ngoại vi thay vì sử dụng eeprom của arduino.

Begin:

Tuổi thọ của eeprom trên các dòng AVR là 100000 lần xóa/ghi, đó là con số do nhà sản xuất đưa ra.

Trong quá trình xóa/ghi dữ liệu, các ô nhớ của eeprom sẽ bị “ bào mòn” dần, một cách tương đối, khi vượt quá giới hạn trên, dữ liệu không còn được lưu trữ chính xác nữa. Các bạn đều biết, dữ liệu mà sai một li, thì kết quả đi một dặm. Do đó quan ngại và giải quyết vấn đề về độ bền của eeprom trên arduino là vô cùng quan trọng.

Hướng giải quyết: Sử dụng ic eeprom.

<ảnh>

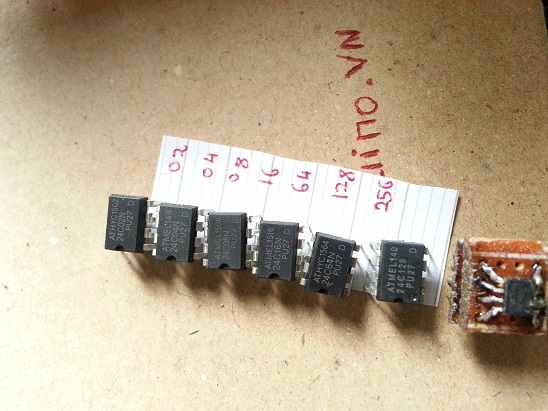
Đầu tiên bạn nên nhớ vài điều này:

* Ô nhớ trên eeprom bị bào mòn dần khi ta thực hiện thao tác ghi dữ liệu hoặc xóa dữ liệu.
* Nếu chỉ đọc dữ liệu thì không ảnh hưởng đến tuổi thọ của eeprom.
* Không sớm thì muộn bạn cũng cần phải biết cách sử dụng eeprom ngoại vi thay vì eeprom có sẵn trên arduino.

Giới thiệu về memory ic – eeprom. 24cxx và cách sử dụng.

Về chức năng và cách thức ghi dữ liệu giống với eeprom trên arduino. Nó thường đi kèm với mọi loại thiết bị số ngày nay để lưu trữ các cài đặt trên thiết bị. ic epprom 24Cxx ở nước ta rất dễ tìm mua và nó cũng rất phổ biến trong làng arduino/mcu trên thế giới. Nó rất rễ sử dụng, giá thành hợp lí.

Dòng ic 24Cxx của atmel sử dụng chuẩn kết nối 2 dây I2c .

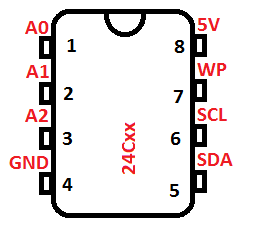
<ảnh>

Có nhiều chủng loại tùy theo dung lượng cho bạn lựa chọn: 24c01, 24c02, 24c04,..vv

Số lần ghi/xóa là 1 triệu lần, gấp cả chục lần arduino.

Vì là ic ngoại vi nên bạn có thể dễ dàng thay thế, nâng cấp, vận chuyển,và cả cộng gộp sức mạnh của các eeprom.

PINOUT:



|  |  |
| --- | --- |
| A0 | 3 chân này để đặt địa chỉ |
| A1 |
| A2 |
| GND | Nối xuống 0v |
| 5V | Nối lên 5v |
| WP (write protect) | Cho phép ghi dữ liệu: Nối chân này xuống 0v |
| Không cho phép ghi dữ liệu: Nối lên 5v. |
| SCL (Serial Clock) | 2 chân cho giao tiếp i2c. |
| SDA (Serial DATA) |

Giao tiếp với arduino;

Chuẩn bị phần cứng:

* 1 Arduino uno r3
* 1 ic eeprom 24Cxx loại bất kì , trong ví dụ là ic 24c04.
* 2 điện trở 4.7k ( hoặc 10k)
* Dây nối, máy hàn…

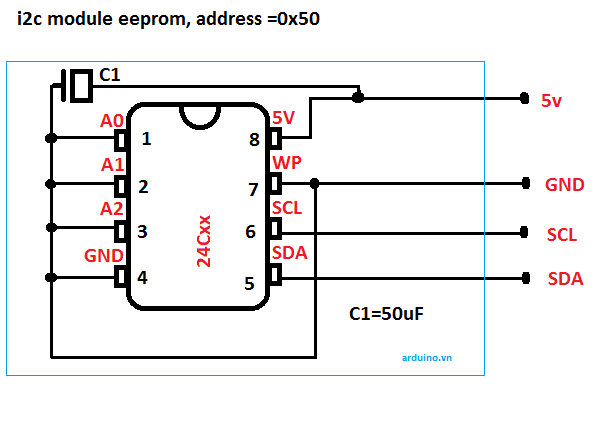
Thư viện:

Đây là thư viện đa năng, nó được dùng cho mọi loại ic eeprom 24cxx.

Tải về tại đây: <link>

Nối dây:

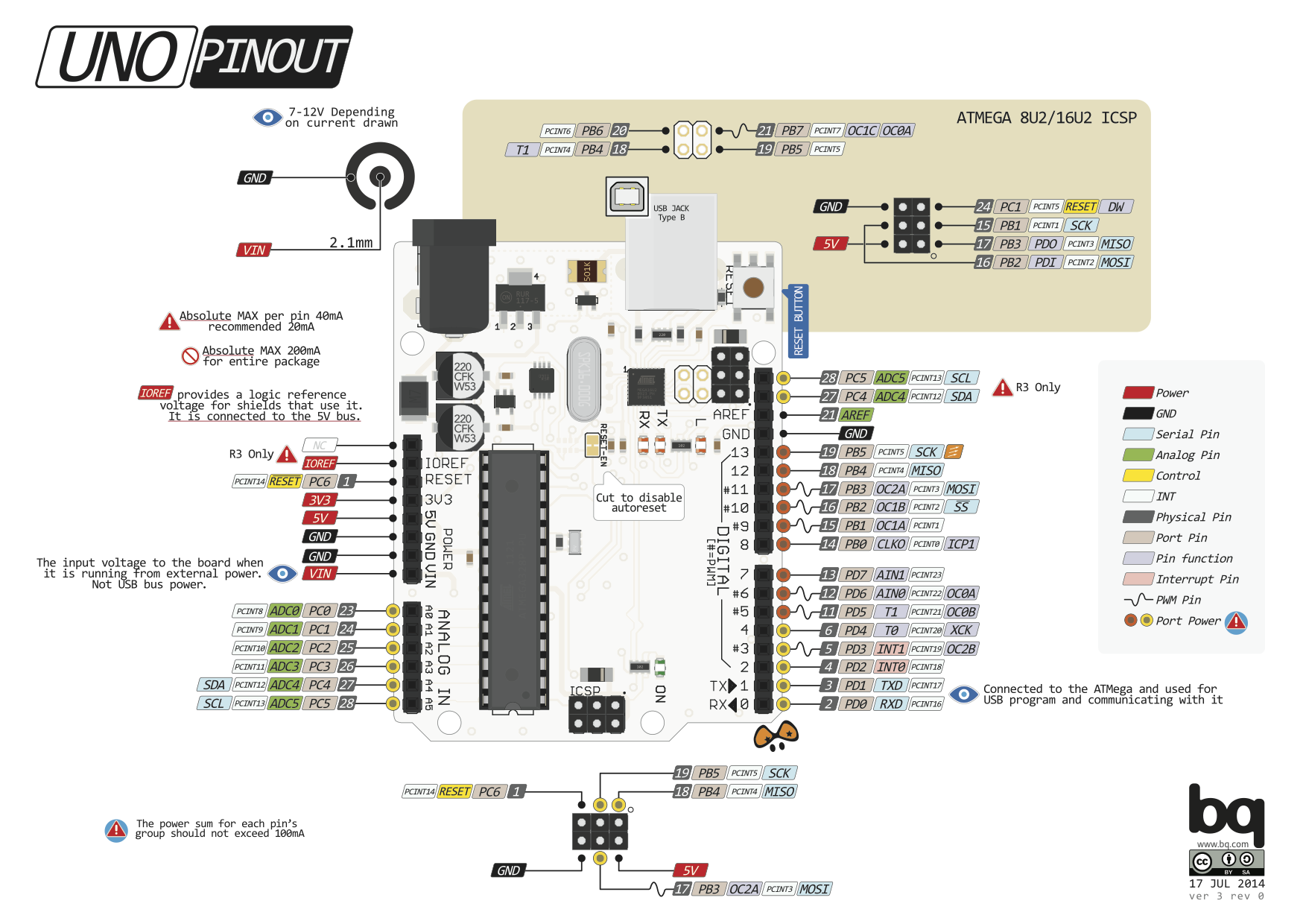
Tạo một module i2c như thế này:



‘

|  |
| --- |
| R1,R2 là hai điện trở pullup để kéo điện áp lên 5v.  C1 là tụ lọc nguồn, bạn có thể không cần nó. |

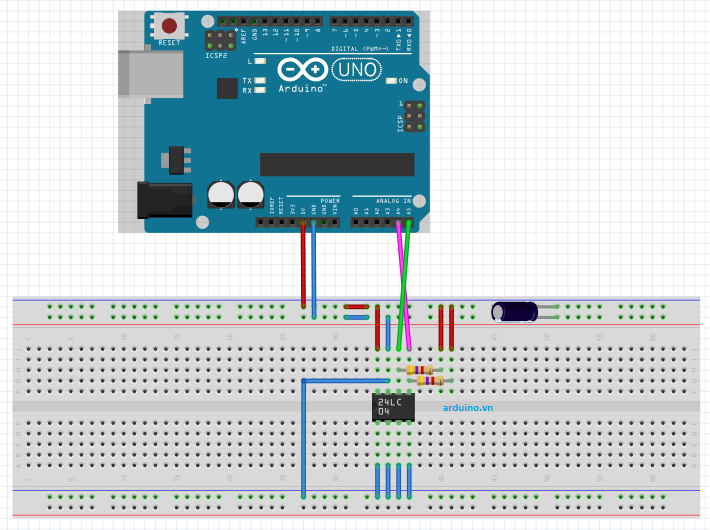
Chúng ta chỉ cần 2 dây dữ liệu để giao tiếp với ic eeprom.



<ảnh >

C1: Với arduino thường.



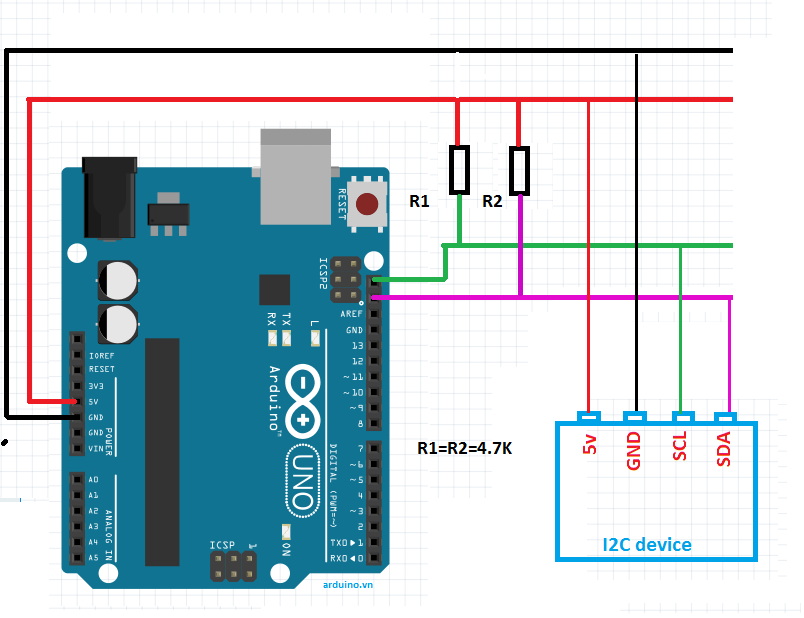


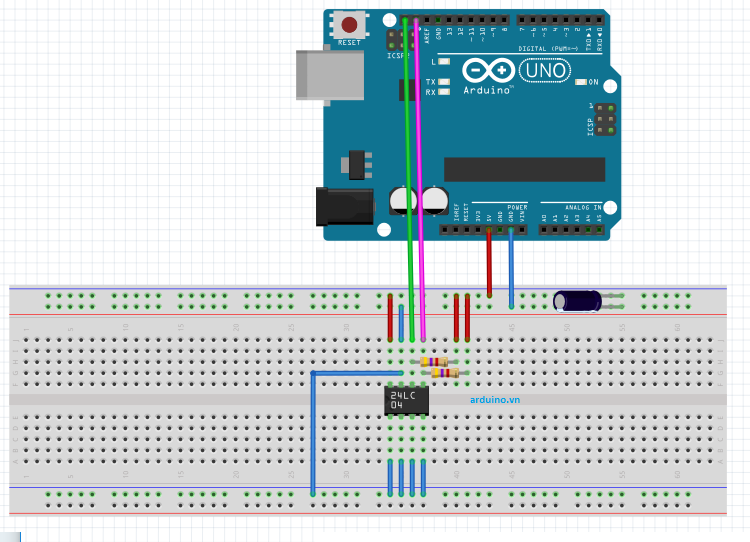
<ảnh>

C2:

Arduino uno r3 có 2 cổng I2c, một cổng lấy từ chân A5,A4 như ví dụ trên, một cổng khác ở vị trí gần nút reset

<ảnh>





<ảnh bread board>

<ảnh nối dây>

Sử dụng thư viện:

Thủ tục khai báo:

<

#include <Wire.h>//b1

#include <Eeprom24Cxx.h>//b2

static Eeprom24C eeprom(4,0x50);// b3

>

B1, B2: cài 2 thư viện. (bạn có thể bỏ đi dòng #include <Wire.h>)

B3:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Static | Eeprom24C | eeprom | 4 | 0x50 |
| Biến tĩnh | Tên class | Tạo và quản lí đối tượng có tên “eeprom” | Mã ic: 04 | Địa chỉ của ic. |

Mã ic nói lên dung lượng của ic eeprom đó (số Page của ô nhớ).

Ta từng biết arduino atm128 có thể lưu 512 biến byte, atm328 có thể lưu 1024 biến byte.

|  |  |  |  |
| --- | --- | --- | --- |
| ***ATmega 48A/PA*** | ***ATmega 88A/PA – 168A/PA*** | ***ATmega328/P*** | **ATmega 640/V-1280/V-1281/V-2560/V-2561/V** |
| *256 bytes* | *512 bytes* | *1 Kbytes* | *4 Kbytes* |

Công thức để tính dung lượng của eeprom là : Số byte=( 1024 \* mã ic) /8=128 \*mã ic.

Bảng 1.

|  |  |  |
| --- | --- | --- |
| Tên ic | Mã (số page) | Dung lượng |
| 24c01 | 1 | 128 bytes |
| 24c02 | 2 | 256 bytes |
| 24c04 | 4 | 512 bytes |
| 24c08 | 8 | 1024 bytes |
| 24c16 | 16 | 2048 bytes |
| 24c32 | 32 | 4096 bytes |
| 24c64 | 64 | 8192 bytes |
| 24c128 | 128 | 16384 bytes |
| 24c256 | 256 | 32768 bytes |

Vân vân…

Địa chỉ của ic :

Tất cả các ic đều có địa chỉ : 0x50. Khi ta nối các chân A0,A1,A2 xuống Low(0v).

Địa chỉ để làm gì?? Nó giúp arduino tìm đến chip eeprom trong giao tiếp i2c thui.

Ghi dữ liệu:

< void write\_1\_byte(unsigned int memory\_address, byte data); >

Ex: eeprom.write(500,123);

// ghi vào byte thứ 500 giá trị bằng 123

Đọc dữ liệu:

<byte read\_1\_byte(unsigned int memory\_address); >

Ex: byte data= eeprom.read(500); Serial.println(data);

Bảng 2

|  |  |  |
| --- | --- | --- |
| Tên ic | Dung lượng | memory\_address |
| 24c01 | 128 bytes | 0->127 |
| 24c02 | 256 bytes | 0->255 |
| 24c04 | 512 bytes | 0->511 |
| 24c08 | 1024 bytes | 0->1023 |
| 24c16 | 2kb | 0->2047 |
| 24c32 | 4kb | 0->4095 |
| 24c64 | 8kb | 0->8191 |
| 24c128 | 16kb | 0->16383 |
| 24c256 | 32kb | 0->32767 |
| 24c512 | 65,5 kb | 0->65635 |

Code test đầy đủ: ic24c04

Các bạn lưu ý nhớ chọn đúng mã ic trên eeprom của bạn nhé.

<

#include <Wire.h>

#include <Eeprom24Cxx.h>

static Eeprom24C eeprom(4,0x50);// mình chọn 24c04.

void setup()

{

Serial.begin(9600);

const int address = 500;

eeprom.write\_1\_byte(address, 123);

delay(5);

Serial.println("Read byte from EEPROM memory...");

byte data = eeprom.read\_1\_byte(address);

Serial.print("Read byte = ");

Serial.print(data);

Serial.println("");

}

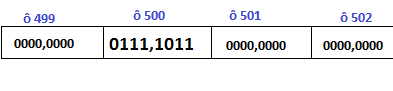
void loop()

{

}

>

Trạng thái ô nhớ:



Cách lưu số lớn hơn kiểu byte ( >255) vào eeprom

Như ví dụ trên, số 123 (0->255) cần 8bit để lưu trữ ,giờ để lưu một số lớn hơn 255 ( từ 16 bit trở lên) thì ta sẽ làm như sau:

Gỉa sử mình cần lưu số 345 vào eeprom ở ô địa chỉ 500.

<

#include <Wire.h>

#include <Eeprom24Cxx.h>

#define type\_ic 4

static Eeprom24C eeprom(type\_ic,0x50);

void setup()

{ Serial.begin(9600);

unsigned int a=345;

}

void loop()

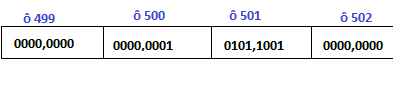
{}

>Đầu tiên ta biết số 345 được biểu diễn nhị phân là 0000,0001,0101,1001.

B1 Ghi :

Tách lấy 8bit cao (0000,0001) lưu vào ô địa chỉ 500

Tách 8 bít thấp (0101,1001) lưu vào ô 501.



B2: đọc

< #include <Wire.h>

#include <Eeprom24Cxx.h>

#define type\_ic 4

static Eeprom24C eeprom(type\_ic,0x50);

void setup()

{ Serial.begin(9600);

unsigned int a=345;

//ghi

byte cao\_8=((a>>8) & 0xFF);

eeprom.write\_1\_byte(500, cao\_8); delay(5);

byte thap\_8=(a & 0xFF);

eeprom.write\_1\_byte(500+1, thap\_8);delay(5);

// đọc

byte doc\_cao\_8, doc\_thap\_8;

doc\_cao\_8=eeprom.read\_1\_byte(500);// 0000,0001

doc\_thap\_8=eeprom.read\_1\_byte(501);// 0101,1001

unsigned int doc\_a;// khai báo// 0000,0000,0000,0000

doc\_a=(doc\_cao\_8<<8);//0000,0001,0000,0000

doc\_a=doc\_a|doc\_thap\_8;//0000,0001,0101,1001

Serial.print(doc\_a);

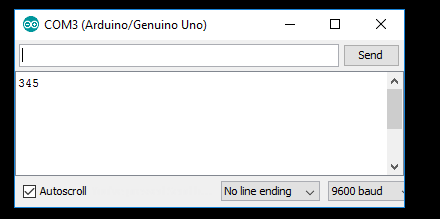
}

void loop()

{

}

>



Một điều cần lưu ý :

* -Số muốn lưu phải là nguyên dương.
* Gỉa sử địa chỉ mà chúng ta muốn lưu là ô 500, khi dùng để lưu một biến có cỡ 2byte thì ô 501 nghiễm nhiên đã bị sử dụng để lưu 8 bit còn lại, do đó, mặc định bạn không được dùng ô 501 để làm địa chỉ lưu , mà phải bắt đầu từ ô 502, bạn nhớ nhé.
* Phải xem sét việc ghi nhiều byte có đang ghi trùng lên các ô khác hay không. Ví dụ ô 500 đã có dũ liệu,Ta dùng ô 488 để lưu 4 byte, việc này sẽ ghi đè dữ liệu vào ô 500, gây sai lệch.
* -Kích cỡ lưu trữ của biến trên arduino và arduino mega khác nhau, Ví dụ kiểu unsigned int trên uno là 2 bytes nhưng trên arduino due là 4 bytes, do đó kích cỡ để lưu cũng khác.

Đây là toàn bộ hàm hỗ trợ ghi theo kiểu dữ liệu mong muốn.

* void write\_1\_byte( unsigned int address, byte data);
* byte read\_1\_byte( unsigned int address);
* uint16\_t read\_2\_byte( unsigned int address);
* void write\_2\_byte( unsigned int address, uint16\_t data);
* void write\_4\_byte( unsigned int address, uint32\_t data);
* uint32\_t read\_4\_byte( unsigned int address);
* void write\_8\_byte( unsigned int address, uint64\_t data);
* uint64\_t read\_8\_byte( unsigned int address);

Như đã đề cập, các kiểu dữ liệu trên các dòng arduino có thể không giống nhau, để biết kiểu dữ liệu (trên arduino của bạn ) chiếm bao nhiêu byte, chúng ta dùng hàm **sizeof();**

Ví dụ: kiểu “**unsigned int**” trên arduino uno r3 chiếm 2 byte.

<

void setup()

{ Serial.begin(9600);

unsigned int a=4002;

Serial.print(sizeof(a));

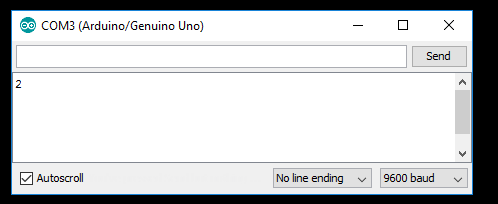
}

void loop()

{

}

>



Code test lưu các số trên arduino uno r3:

<

#include <Wire.h>

#include <Eeprom24Cxx.h>

#define type\_ic 4

static Eeprom24C eeprom(type\_ic,0x50);

void setup()

{ Serial.begin(9600);

byte a=255;

unsigned int b=62770;

unsigned long c=4000000000; //4 tỷ

eeprom.write\_1\_byte(0, a);

eeprom.write\_2\_byte(10, b);

eeprom.write\_4\_byte(40, c);

Serial.println(eeprom.read\_1\_byte(0));

Serial.println(eeprom.read\_2\_byte(10));

Serial.println(eeprom.read\_4\_byte(40));

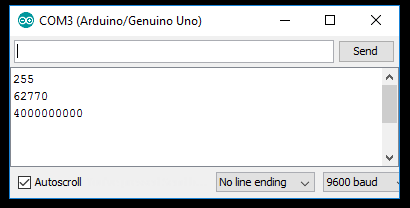
}

void loop()

{

}

>



Thư viện này còn có thể làm điều tương tự trên eeprom của arduino.

Bạn truy cập các phương thức với tên thực thể là: AVR\_EEPROM.

Đồng thời không khai báo ic eeprom, thư viện sẽ tự hiểu là chúng ta đang muốn dùng eeprom trên arduino.

* void write\_1\_byte( unsigned int address, byte data);
* byte read\_1\_byte( unsigned int address);
* uint16\_t read\_2\_byte( unsigned int address);
* void write\_2\_byte( unsigned int address, uint16\_t data);
* void write\_4\_byte( unsigned int address, uint32\_t data);
* uint32\_t read\_4\_byte( unsigned int address);
* void write\_8\_byte( unsigned int address, uint64\_t data);
* uint64\_t read\_8\_byte( unsigned int address);

<

#include <Wire.h>

#include <Eeprom24Cxx.h>

void setup()

{ // viết và ghi dữ liệu lên eeprom của arduino

Serial.begin(9600);

byte a=255;

unsigned int b=62770;

unsigned long c=4000000000; //4 tỷ

AVR\_EEPROM.write\_1\_byte(0, a);

AVR\_EEPROM.write\_2\_byte(10, b);

AVR\_EEPROM.write\_4\_byte(40, c);

Serial.println(AVR\_EEPROM.read\_1\_byte(0));

Serial.println(AVR\_EEPROM.read\_2\_byte(10));

Serial.println(AVR\_EEPROM.read\_4\_byte(40));

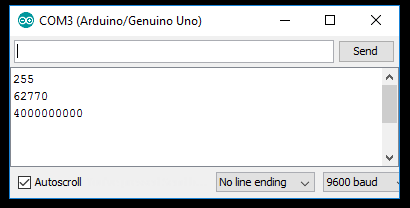
}

void loop()

{

}

>



5: Hoàn thiện kĩ năng .

100.000 lần ghi/xóa của arduino hay 1 triệu lần của ic 24Cxx đều là con số lớn. Lớn đến vậy mà chuyện ghi/xóa vô tội vạ cũng khiến bạn phải thường xuyên thay thế 1 ic eeprom mới.

Mình xin gợi ý một vài thủ thuật để nâng cao tuổi thọ và tốc độ cho ic eeprom:

* Vì bạn có thể đọc thoải mái dữ liệu mà không làm giảm tuổi thọ, nên trước khi ghi, bạn hãy kiểm tra xem dữ liệu đưa vào, nếu trùng khớp với dữ liệu cũ thì không cần ghi.( Cái này thư viện đã hỗ trợ rồi).
* Nếu được, bạn hãy thường xuyên thay đổi vị trí ghi, để san sẻ số lần ghi cho các ô nhớ khác, nó giúp ta tránh việc phải vứt cả ic eeprom chỉ vì một hai ô nhớ bị hỏng. (Đây gọi là phân mảnh bộ nhớ)
* PIN số 7 (wp) trên ic giúp ta ngăn việc ghi dữ liệu bằng cách nối chân này lên HIGH (5v).
* Để tốc độ được nhanh, nếu phải liên tục đọc lại dữ liệu ở cùng một ô nhớ trên eeprom, bạn hãy khai báo một biến, gán cho nó giá trị rồi chỉ đọc trên biến đó thôi.
* Phải delay ít nhất 5 mili giây(ms) sau khi ghi để đảm bảo dữ liệu được lưu hoàn toàn.
* Bạn hãy lưu tạm dữ liệu vào ram ( khai báo một biến rồi gán dữ liệu cho biến đó), khi thật sự cần thiết thì mới cho lưu eeprom. Cách này thường dùng để tránh mất mát dữ liệu đang tính toán khi có sự cố mất điện. Cái này bạn có thể tìm hiểu thêm.

3: Bạn muốn dung lượng cao hơn ??

Tất nhiên là có ic cho bạn 24lc567 , 24lc1024 , vv..

Tuy nhiên, các ic này sử dụng giao tiếp SPI 5 dây, bạn hãy tìm hiểu hêm nhé.

4: ic eeprom 24Cxx tháo máy !!!

Do mình viết lại thư viện, để đảm bảo tính chất lượng , nên đã mình đặt mua các loại ic khác nhau về để test trước khi đăng lên arduino.vn. Tuy nhiên trước đó, mình đã từng vọc loại ic eeprom này ở khá nhiều thiết bị khác nhau

Tivi có chú 24c04, tivi nào cũng có cả, mình nghĩ chúng để lưu các cài đặt của người dùng .

<ảnh>

Cái main tivi cũ kĩ này cũng có cái để vọc đấy^^

|  |  |
| --- | --- |
|  |  |

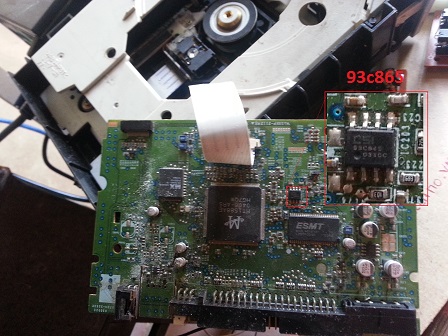
Trên thanh ram có chú 24C02.

<ảnh>

Trên card mạng có để lưu địa chỉ MAC, bạn nào có ý định thay địa chỉ MAC để thành HACKER thì đúng bài rồi ..hehe (Cái này mình không khuyến khích đâu nhé).

<ảnh>

Trong cái ổ VCD của máy tính có chú 93c865 cơ ^^



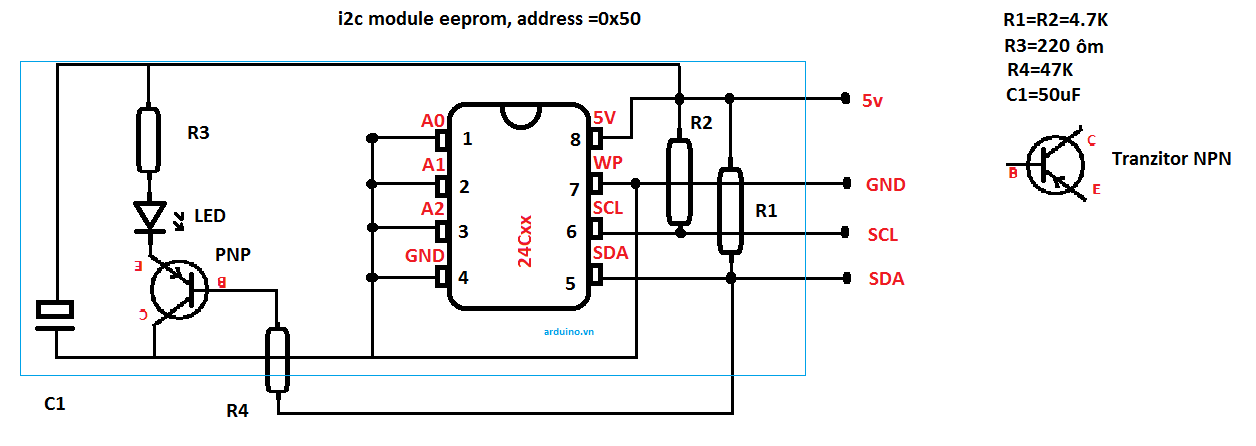
<ảnh>

Module Ic thời gian thực.

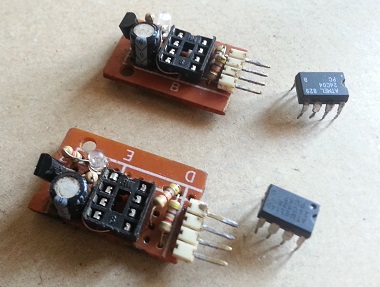
Tất nhiên các eeprom trên đều đã có “tuổi”, nếu không tin tưởng nên mua mới ic thay vì dùng lại như trên.

6: Tự chế một module.

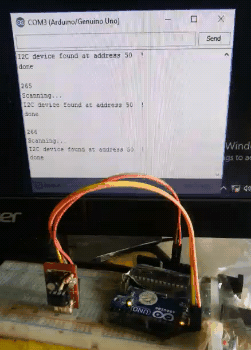
Vì ic nhỏ gọn, nên có thể bạn sẽ cần dùng nó giống như một usb eeprom. Hãy chế theo cách của bạn.

* 

ở phiên bản này, mình có thêm led và tranzitor npn ,đèn sẽ sáng khi có tín hiệu mức thấp, báo là đang có truyền dẫn data.

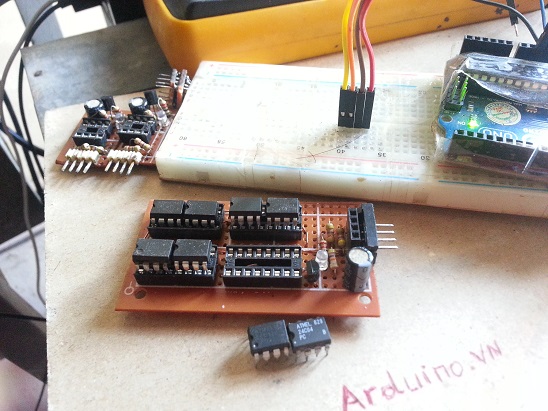


<ảnh board>’<ảnh gif>



8: BUS IC

Dòng ic 24cxx có địa chỉ nằm trong khoảng 1010A2 A1A0 ( 0x50->0x57), có nghĩa là ta có thể có tới 8 ic eeprom dùng chung một bus i2c. Vậy bạn có muốn độ eeprom thành super eeprom không??. Hãy đón đọc bài tiếp theo nhé.



<ảnh>

5: Cái kết có hậu

Có lẽ sau khi ta sử dụng được ic này rồi, hẳn sẽ thấy nó không quá khó. Đặc biệt hơn, chúng ta sẽ không còn phụ thuộc vào cái eeprom của arduino, và tất nhiên là bạn đã lên level rồi đấy.

Hi vọng bài viết hữu ích và bạn thấy thích bài viết này.

<Thái Sơn>

BÀI 2:

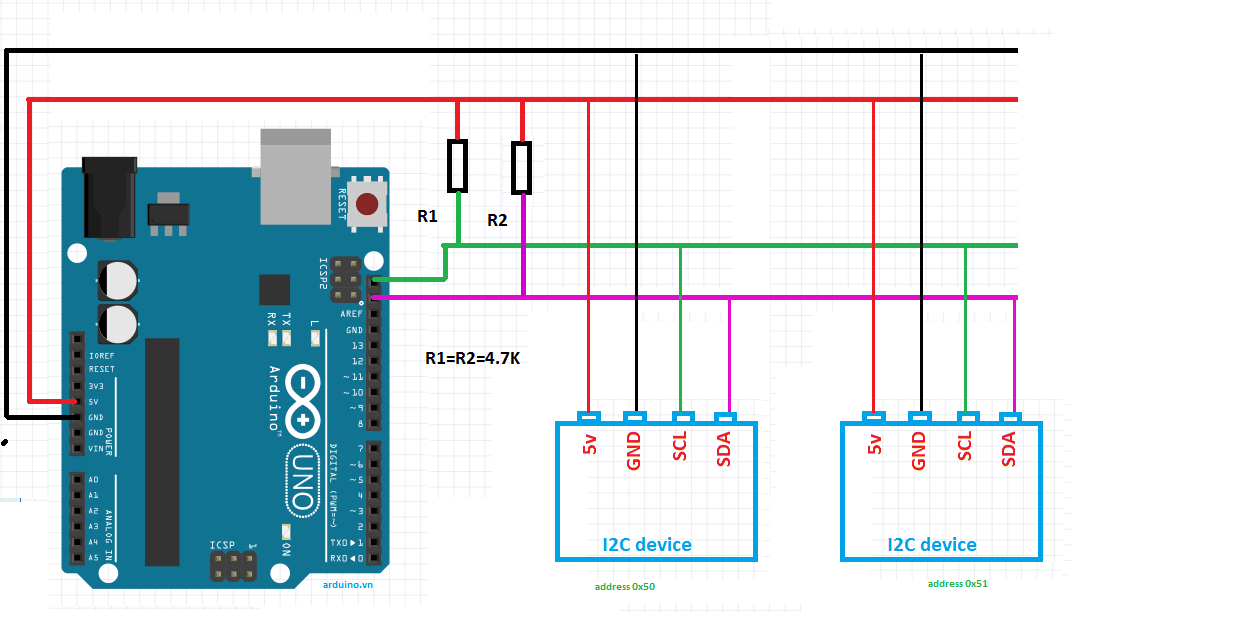
Sử dụng nhiều ic cùng lúc .

Trên 1 Bus i2c (2 dây) có thể có tới 128 thiết bị khác nhau cùng dùng chung đường dây.

Lúc này, địa chỉ của mỗi thiết bị sẽ giúp arduino (master) tìm đến và giao tiếp truyền nhận với thiết bị đó (slave). Tất nhiên địa chỉ phải của mỗi thiết bị phải khác nhau.

Dùng chung đường dây có nghĩa là các thiết bị (modul) sẽ nối song song với nhau trên đường bus chung, điều này làm tiết kiệm đáng kể lượng dây kết nối với arduino/mcu.

<ảnh>



Bây giờ chúng ta bàn về địa chỉ.

Địa chỉ của ic eeprom trên bus do người dùng thiết lập bằng cách đấu nối các chân A0, A1, A2 với mức điện áp 0v/5v .

Địa chỉ xác định bởi 7 bit:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | A2 | A1 | A0 |

3 bit cuối là trạng thái của 3 pin .

* Bằng 1: Nối pin đó lên điện áp High (5v)
* Bằng 0: :Nối pin đó xuống điện áp Low (0v)

Nhà sản xuất đã giới hạn khả năng thiết lập địa chỉ với từng ic:

Bảng 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| Tên ic | A2 | A1 | A0 |
| 24c01 | **X** | **X** | **X** |
| 24c02 | **X** | **X** | **X** |
| 24c04 | ***V*** | ***V*** | **X** |
| 24c08 | ***V*** | **X** | **X** |
| 24c16 | **X** | **X** | **X** |
| 24c32 | ***V*** | ***V*** | ***V*** |
| 24c64 | ***V*** | ***V*** | ***V*** |
| 24c128 | ***V*** | ***V*** | ***V*** |
| 24c256 | ***V*** | ***V*** | ***V*** |

Với trường hợp **X**:

Pin không có khả năng thay đổi địa chỉ của ic

Khi đó giá trị tại địa chỉ này mặc định bằng Low(0) bạn được phép để hở chân này, hoặc nối xuống nguồn 0v . Và không được nối chân này lên High(5v).

Với trường hợp ***V:***

Pin khả dụng để thay đổi địa chỉ eeprom bằng cách nối dây.

* Bằng 1: Nối pin đó lên điện áp High (5v)
* Bằng 0: :Nối pin đó xuống điện áp Low (0v)

Trường hợp chung để suy ra địa chỉ: 1010A2A1A0 rồi đổi sang hex.

Bảng 2.1

|  |  |  |  |
| --- | --- | --- | --- |
| A2 | A1 | A0 | Address (hex)= |
| 0 | 0 | 0 | 0x50 |
| 0 | 0 | 1 | 0x51 |
| 0 | 1 | 0 | 0x52 |
| 0 | 1 | 1 | 0x53 |
| 1 | 0 | 0 | 0x54 |
| 1 | 0 | 1 | 0x55 |
| 1 | 1 | 0 | 0x56 |
| 1 | 1 | 1 | 0x57 |

Ví dụ:

Tất cả các ic đều có địa chỉ : 0x50., địa chỉ này thường dùng như một địa chỉ mặc định khi chỉ muốn dùng 1 ic trên bus.

XÉT IC 24C04, 24C08, 24C16.

Như ta đã biết mỗi thiết bị trên đường bus chỉ được phép mang một và chỉ một địa chỉ duy nhất.

 Chạy code i2\_scanner ( có trong mục example của thư viện, các bạn nối mạch như trên)Kiểm tra địa chỉ của các ic ta được kết quả sau.

Bảng 3.1

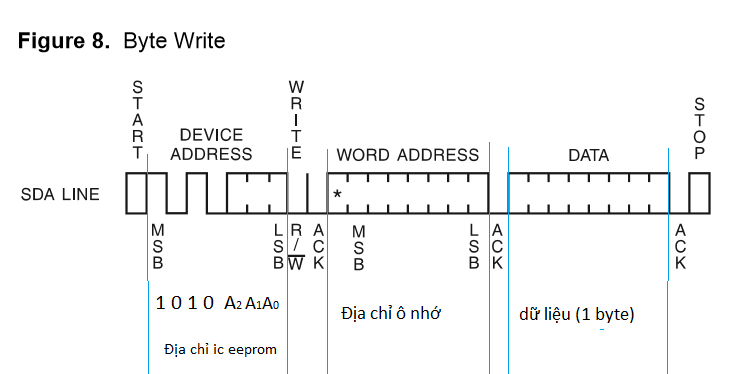
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A2 | A1 | A0 | Địa chỉ thu được hiện lên trên cửa sổ Serial. |
| 24c04 | ***0*** | ***0*** | 0 | 0x50 và 0x51 |
| ***0*** | ***1*** | **0** | 0x52 và 0x53 |
| ***1*** | ***0*** | **0** | 0x54 và 0x55 |
| ***1*** | ***1*** | **0** | 0X56 và 0x57 |
| 24c08 | ***0*** | **0** | **0** | 0x50,0x51, 0x52, 0x53 |
|  | ***1*** | **0** | **0** | 0x54, 0x55, 0x56, 0x57 |
| 24c16 | **0** | **0** | **0** | Từ 0x50 đến 0x57 |

Thật ngạc nhiên khi trên một ic lại có thể có tới nhiều hơn 1 địa chỉ. Vậy ta nên chọn địa chỉ nào để giao tiếp arduino với ic eeprom. ???

Giải thích nguyên nhân:

Do sự nâng cấp về dung lượng, ic được phân ra làm các trang ô nhớ (Page), mỗi trang có dung lượng 256 ô nhớ. 24c04 có 2 Page, 24c08 có 4 Page, 24c16 có 8 Page.

Do đó chỉ với 1 byte địa chỉ ô nhớ sẽ không thể truy cập đến dung lượng lớn hơn. Người ta giải quyết bằng cách sử dụng chân A0 (hoặc A1,A2) làm chân chọn Page.



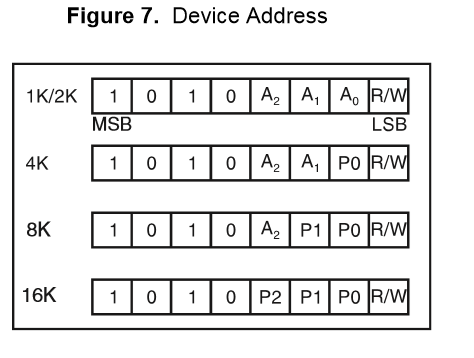
Ví dụ ic24c04:

Với A0=0, địa chỉ truy cập ô nhớ chạy từ 0->255 thì ô nhớ trên ic là từ 0->255 của Page 1.

Với A0=1, địa chỉ truy cập ô nhớ chạy từ 256->511 thì ô nhớ trên ic là từ 0->255 của Page 2.

Cứ như thế người ta sử dụng nốt các chân A1, A2 làm các chân chọn Page để tăng dung lượng mà vẫn đảm bảo tốc độ giao thức.

Có nghĩa bảng địa chỉ ic eeprom sẽ được hiểu như sau:



Điều này cũng đã được nói rõ trong datasheet .

Đây cũng là lời giải thích cho vấn đề đã nêu: Một ic có thể mang nhiều địa chỉ.

Vậy thì ta sẽ chọn địa chỉ bằng cách cho P=0, như sau:

Bảng 4.1

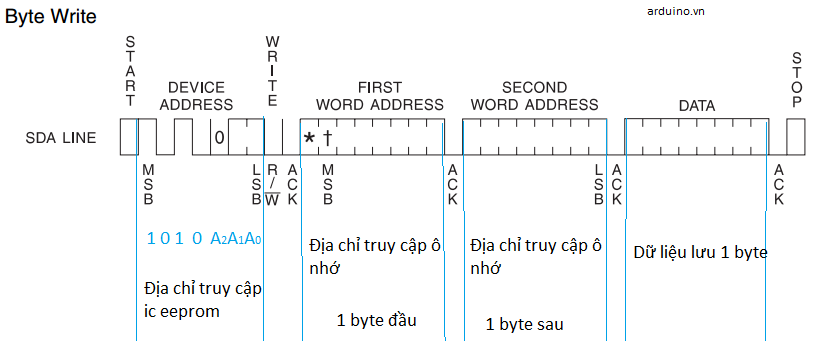
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A2 | A1 | A0 | Địa chỉ thật của ic |
| 24c04 | ***0*** | ***0*** | 0 | 0x50 |
| ***0*** | ***1*** | **0** | 0x52 |
| ***1*** | ***0*** | **0** | 0x54 |
| ***1*** | ***1*** | **0** | 0X56 |
| 24c08 | ***0*** | **0** | **0** | 0x50 |
|  | ***1*** | **0** | **0** | 0x54 |
| 24c16 | **0** | **0** | **0** | 0x50 |

* Bạn nên nhớ, mỗi Slave chỉ có một và chỉ một địa chỉ để truy cập mà thôi.
* Không nên để cho các Slave có địa chỉ giống nhau trên cùng một bus.

Về cách thức truy cập của các ic trên cũng khác nhau, bạn có thể theo dõi trong thư viện, mình đã chú thích rất rõ .

Ic 24c32, 24c64, 24c128, 24c256.

Dung lượng ic ngày càng tăng, chân A2,A1,A0 cũng đã dùng hết. Người ta lại tiến hành tăng độ dài chuỗi ô địa chỉ nhớ lên 2 byte.



Lúc này dung lượng có thể lên tới tới 256\*256 ô nhớ.

Còn các chân A2,A1,A0 được quay về chức năng nguyên thủy của nó là cài địa chỉ cho ic.

Không còn chân chọn Page, mỗi ic ( 32,64,128,256)có thể có tới 8 địa khác nhau bằng cách thiết lập như thông thường.

Bảng 5.1

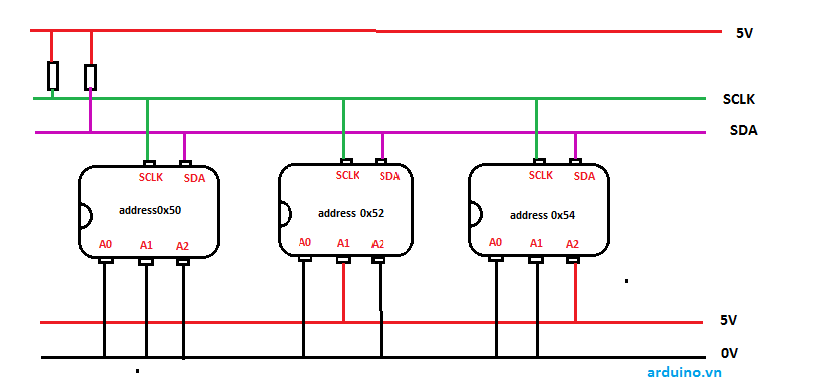
|  |  |  |  |
| --- | --- | --- | --- |
| IC 24c32, 24c64, 24c128, 24c256. | | | |
| A2 | A1 | A0 | Address (hex)= |
| **0** | **0** | **0** | 0x50 |
| **0** | **0** | **1** | 0x51 |
| **0** | **1** | **0** | 0x52 |
| **0** | **1** | **1** | 0x53 |
| **1** | **0** | **0** | 0x54 |
| **1** | **0** | **1** | 0x55 |
| **1** | **1** | **0** | 0x56 |
| **1** | **1** | **1** | 0x57 |

Ok, chúng ta đi vào ví dụ sử dụng:

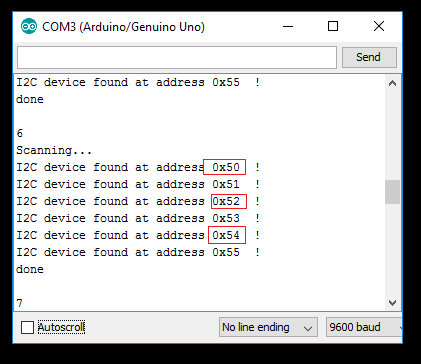
Gỉa sử mình muốn sử dụng 3 ic 24c04 trên cùng một đường bus:

Nhìn vào bảng 4.1, mình sẽ nhặt ra 3 địa chỉ hợp lệ là 0x50 (cho ic1), 0x52 (cho ic2), 0x54 (ic3).





Kiểm tra bằng i2c\_scanner :



Cái mình đóng khung đỏ chính là địa chỉ ta đã thiết lập bằng phần cứng của 3 ic,( địa chỉ ảo 0x51 , 0x53,0x55, là của Page 2 có trên mỗi ic. Chúng ta không quan tâm đến nó).

<

#include <Wire.h>

#include <Eeprom24Cxx.h>

static Eeprom24C ic1(4,0x50);

static Eeprom24C ic2(4,0x52);

static Eeprom24C ic3(4,0x54);

void setup()

{

Serial.begin(9600);

//ghi

const int address = 200;// lưu vào ô 200

ic1.write\_1\_byte(address,10); delay(5);

ic2.write\_1\_byte(address,20); delay(5);

ic3.write\_1\_byte(address,30); delay(5);

//đọc

Serial.println("Read byte from EEPROM memory...");

byte d1,d2,d3;

d1=ic1.read\_1\_byte(address);

d2=ic2.read\_1\_byte(address);

d3=ic3.read\_1\_byte(address);

//in ra

Serial.println(d1);

Serial.println(d2);

Serial.println(d3);

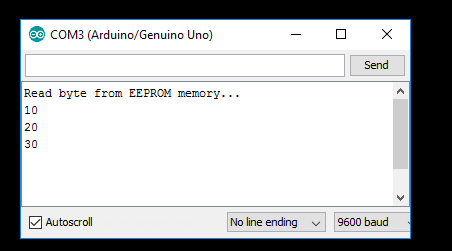
}

void loop()

{

}

>



**Vẫn giữ nguyên 3 ic , mình muốn dùng thêm ic 24c02, 24c08, 24c16, 24c256, 24c64, 24c128.**

Vì chỉ có tối đa 8 ic eeprom trên 1 bus nên mình sẽ lập 1 bảng để phân bố sao cho các địa chỉ thuộc địa chỉ cho phép và khác các địa chỉ còn lại.

B1: Bỏ ic 24c02 vì đã quá số lượng cho phép.

B2: Nhìn vào bảng 4.1 và 5.1 , mình cần chọn ra các địa chỉ phù hợp nhất:

|  |  |
| --- | --- |
| 24c16 | 0x50. Đây cũng là địa chỉ duy nhất của ic nên phải ưu tiên chọn trước |

|  |  |
| --- | --- |
| 24c08 | 0x54. (0x50 đã bị dùng) |

|  |  |
| --- | --- |
| 24c04 (ic1) | 0x52.(50,54 đã bị dùng) |
| 24c04 (ic2) | 0x56.( 50,52,54, đã bị dùng) |

Còn 4 địa chỉ chưa dùng: 0x51,0x53, 0x55, 0x57

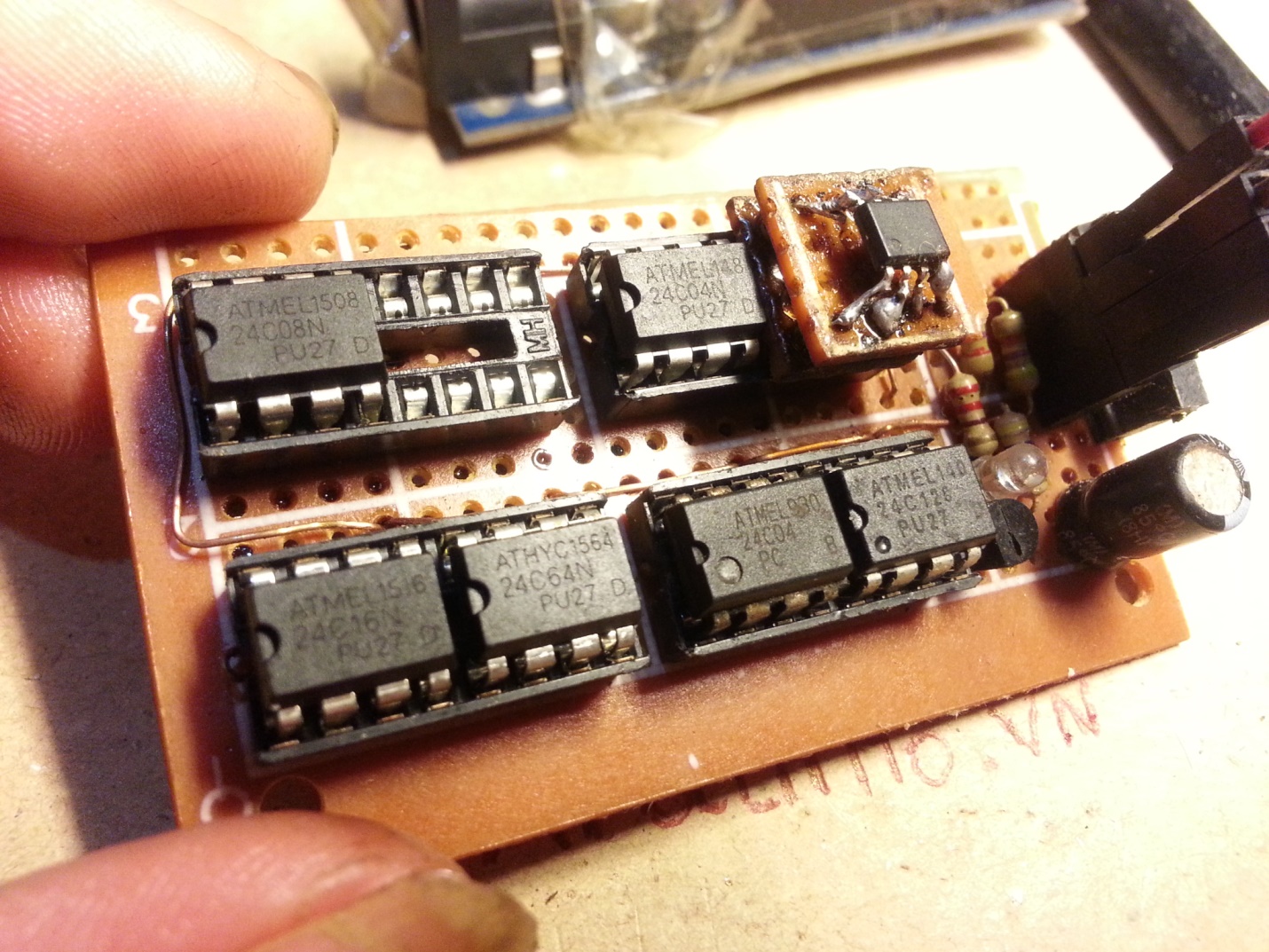
|  |  |
| --- | --- |
| 24c064 | 0x51 |
| 24c128 | 0x53 |
| 24c256 | 0x57 |

Chúng ta còn dư 1 ic 24c04(ic3) do không còn địa chỉ phù hợp.

B3: ráp mạch, tùy vào địa chỉ mà các bạn nối chân A cho phù hợp.

Do biết được đặc điểm là chỉ có thể có tối đa 8 ic nên mình đã làm một Sheil như thế này,

Có 8 đế ic, mỗi đế có sãn địa chỉ khác nhau (0x50->0x57).



<ảnh>

Upcode test:

<

#include <Wire.h>

#include <Eeprom24Cxx.h>

static Eeprom24C ic\_16(16,0x50);

static Eeprom24C ic\_64(64,0x51);

static Eeprom24C ic\_4\_1(4,0x52);

static Eeprom24C ic\_128(128,0x53);

static Eeprom24C ic\_8(8,0x54);

static Eeprom24C ic\_4\_2(4,0x56);

static Eeprom24C ic\_256(256,0x57);

// quản lí các đối tượng bằng mảng.

static Eeprom24C ic[7]={ic\_16, ic\_64, ic\_4\_1, ic\_128, ic\_8,ic\_4\_2, ic\_256};

void setup()

{

Serial.begin(9600);

//ghi

const int address = 100;// lưu vào ô 100

for( byte i=0; i<7; i++){

ic[i].write\_1\_byte( address, (i+1)\*10);

delay(5);

}

//đọc

Serial.println("Read byte from EEPROM memory...");

for( byte i=0; i<7; i++){

byte d;

d=ic[i].read\_1\_byte( address);

//in ra

Serial.println(d);

}

}

void loop()

{

}

>

Kết quả:



Nhìn vào kết quả, ta có thể khẳng định , arduino đã tìm đến đúng ic mong muốn. Không cái nào trùng cái nào.

Lời khuyên ,Nếu bạn là beginner, bạn cảm thấy việc tìm địa chỉ gây khó khăn, mình khuyên bạn chỉ nên sử dụng ic 24c32, 24c64, 24c128, 24c256 (giá mua của nó chỉ cao hơn một chút xíu thôi, bạn hãy đầu tư nhé). Việc cài đặt địa chỉ rất dễ dàng. Bạn cũng chưa cần cố gắng hiểu đến những gì mình đã viết về ic khác (24c16 trở xuống).

Đó là bước làm quen tốt nhất.

Kết >

HEHE, từ mục tiêu tìm hiểu cái eeprom trong tivi vứt xó đến khi mình viết được bài viết này là cả một quãng đường đầy thú vị. Ở nước ta, việc sử dụng ic này đã phổ biến rồi. Tuy vậy, tài liệu hướng dẫn vẫn rất hạn chế. Thậm chí là không ai muốn đả động đến phần sử dụng cùng lúc nhiều ic trên bus do có một chút nhập nhằng về cách thiết lập địa chỉ, các bài nước ngoài cũng chỉ nói đến 8 con ic 256 giống nhau, chứ chưa đề cập 8 con khác nhau như bài của mình.

Vấn đề về ic 24cxx cũng đã hết.

Tạm kết tại đây, rất mong sự ủng hộ của các bạn và hi vọng bạn thấy bài viết này hữu ích.

<Thái Sơn>

Bài được đăng lên trang arduino.vn