ĐIỀU KHIỂN THIẾT BI

QUA ĐƯỜNG DÂY ĐIỆN THOẠI

2008

ĐỀ TÀI VI XỬ LÍ

VÕ HOÀNG MẠNH HÙNG

NGUYỄN HỮU CƯỜNG

GVBM: PHẠM THẾ DUY

Đ05VTA1-ptit

1/1/2008

MỤC LỤC

DANH SÁCH HÌNH VẼ TRONG BÁO CÁO

I. MỤC ĐÍCH CỦA ĐỀ TÀI

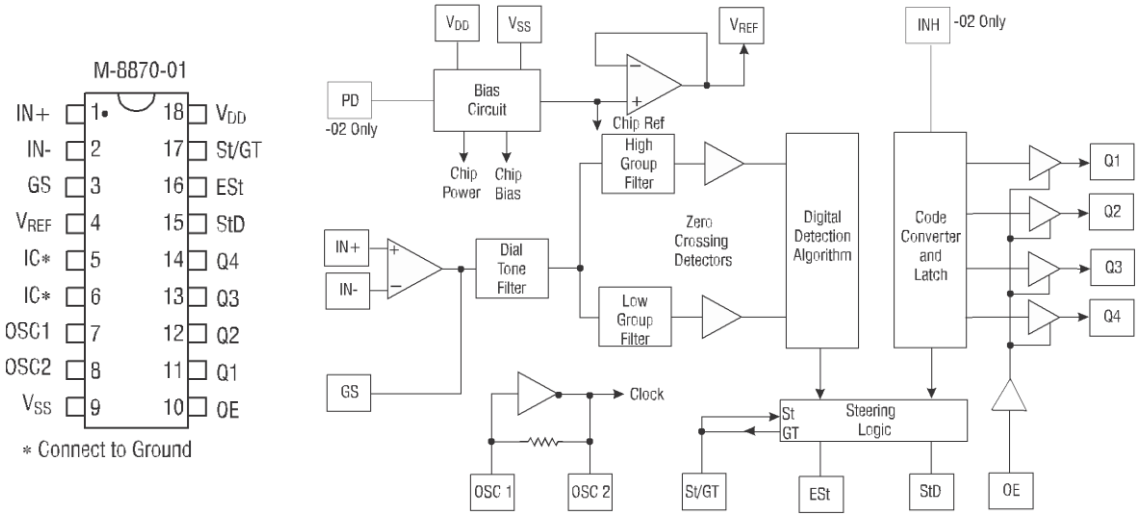
Điều khiển thiết bị qua đường dây điện thoại là một đề tài mà không ít người đã từng quan tâm và theo đuổi. Đây là một đề tài hay và có rất nhiều giá trị trong thực tế, dùng để điều khiển thiết bị từ xa ở mọi khoảng cách , miễn sao điện thoại của bạn có thể goi tới được địa điểm có gắn thiết bị.

Trong đề tài này , chúng tôi tạo ra một hệ thống điều khiển động cơ (dùng đóng, mở cửa hay chạy robot chẳng hạn). Hệ thống này được kiểm soát bằng mật mã có thể thay đổi được và có sự phản hồi tới người dùng bằng tiếng nhạc.

II. LINH KIỆN CẦN THIẾT

1. 8870

8870 là IC giải mã DTMF . Nó nhận tín hiệu DTMF từ đường dây điện thoại sau đó giải mã thành 1 số 4 **bit** tương ứng với tín hiệu DTMF mà nó nhận được.



Hình 1-Cấu tạo 8870

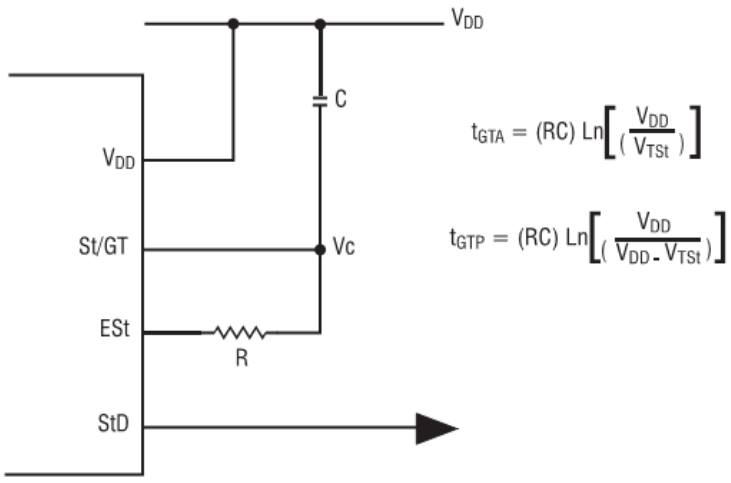
Dùng 8870, ta có thể biết được người dùng đã bấm số nào trên bàn phím điện thoại , từ đó đưa ra tín hiệu điều khiển phù hợp.

8870 chứa 2 bộ lọc thông dải , dùng để tách cặp tone DTMF nhận được thành 1 tone thuộc nhóm tone cao và 1 tone thuộc nhóm tone thấp. Bộ giải mã số nằm trong 8870 sẽ xác nhận cặp tone DTMF nhận được , nếu cặp tone này tồn tại trong 1 khoảng thời gian đủ dài định trước thì 4 **bit** mã tương ứng của nó sẽ được chuyển đến các ngõ ra Q1→Q4, đồng thời 1 ngắt ở chân StD được sinh.

F_{LOW}	F_{HIGH}	Key (ref.)	Q4	Q3	Q2	Q1
97	1209	1	0	0	0	1
697	1336	2	0	0	1	0
697	1477	3	0	0	1	1
770	1209	4	0	1	0	0
770	1336	5	0	1	0	1
770	1477	6	0	1	1	0
852	1209	7	0	1	1	1
852	1336	8	1	0	0	0
852	1477	9	1	0	0	1
941	1336	0	1	0	1	0
941	1209	S	1	0	1	1
941	1477	#	1	1	0	0
697	1633	A	1	1	0	1
770	1633	B	1	1	1	0
852	1633	C	1	1	1	1
941	1633	D	0	0	0	0

Hình 2 - Mã DTMF

Mạch lái



Hình 3-Mạch lái

Trước khi đưa ra ngoài 4 **bit** mã tương ứng với cặp tone nhận được, bộ nhận kiểm tra khoảng thời gian tồn tại của tín hiệu . Việc kiểm tra dựa vào thời hằng RC gắn ngoài qua ngõ Est.

Est lên cao khi có 1 cặp tone DTMF được nhận diện . ESt còn lên cao chừng nào cặp tone này còn tồn tại . ESt lên cao làm Vc tăng lên . Sau khoảng thời gian t_{GTP}, Vc vượt mức ngưỡng V_{TSt} (khoảng 2,2→2,5V), 4**bit** mã tương ứng sẽ được chuyển đến chốt ngõ ra. Lúc này chân GT kích hoạt l àm Vc tiếp tục tăng đến V_{DD}. Cuối cùng, sau khi dừng 1 thời gian ngắn để chốt ngõ ra ổn định, cờ ngõ ra mạch lái (steering output flag, StD) lên cao, báo hiệu rằng cặp tone nhận được đã được chấp nhận và 4 **bit** mã tương ứng đã được chuyển đến ngõ ra.

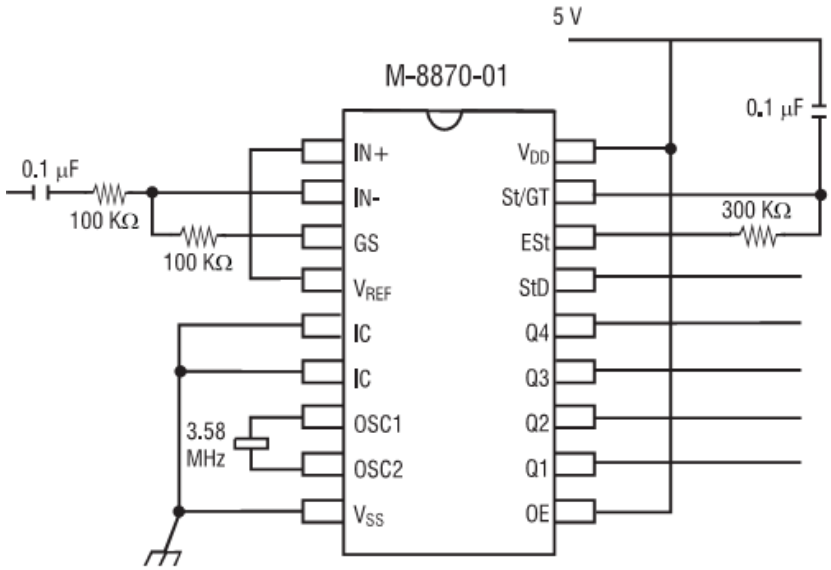
Mạch lái hoạt động ở chế độ trái ngược để xác minh những đoạn ngắt quãng giữa các tín hiệu. Như vậy, bộ nhận sẽ bỏ qua những sự ngắt quãng tín hiệu quá ngắn cũng như những tín hiệu quá ngắn.

Khả năng này cùng với tính năng cho phép chọn thời hằng bên ngoài cho phép người thiết kế có thể điều chỉnh hệ thống 1 cách phong phú phù hợp mục đích hệ thống

Cấu hình ngõ vào: có 2 loại cấu hình ngõ vào

+ Cấu hình gồm 1 ngõ vào, ta sẽ sử dụng cấu hình này

+ Cấu hình gồm cặp ngõ vào sai biệt



Hình 4-Cấu hình ngõ vào

Ngõ vào 8870 bao gồm 1 opamp có ngõ vào sai biệt và 1 nguồn phân cực Vref để phân cực ngõ vào của bộ khuếch đại ở VDD/2. R hồi tiếp ở ngõ ra của opamp (GS) dùng để điều chỉnh độ lợi của tín hiệu vào.

Thạch anh 3.58MHz dùng để tạo dao động cho 8870.

2. ***24c04***

24C04 là bộ nhớ nối giao tiếp theo chuẩn I 2C. Kết nối của loại bộ nhớ này rất đơn giản, lại nhỏ gọn. Tuy nhiên, phần giao tiếp của nó với vi xử lí lại phức tạp.

24C04 giao tiếp với vi xử lí bằng 2 ngõ SDA và SCL. SCL tạo xung nhịp còn SDA là đường dữ liệu.

Chức năng của 24c04:

+ Truyền dữ liệu qua bus 2 dây

+ Giao tiếp với µP, trong đó µP là thiết bị chủ, 24C04 là thiết bị nhận. µP sẽ quyết định chế độ làm việc.

Các đặc tính kĩ thuật của 24C04:

- Dung lượng 512 bytes (4Kb)
  ⇨ 24c04 có thể lưu được 512 mật mã trở lên, nếu lập trình tốt.
- Tần số 400kHz đối với nguồn cung cấp 5V
- Chu kì clock SCL tối thiểu là 2,5µs
- Thời gian clock SCL lên cao tối thiểu 0.6µs
- Thời gian SCL xuống thấp tối thiểu 1.2µs
  ⇨ Vi xử lí phải chờ trong thời gian đủ để duy trì trạng thái
- Thời gian đổi trạng thái lên cao tối đa 0.3µs
- Thời gian đổi trạng thái lên cao 0.3µs
  ⇨ có thể đổi trạng thái ngay, không phải chờ
- Thời gian BUS rảnh trước khi truyền dữ liệu mới (sau lệnh STOP) tối thiểu 1.2µs
- Thời gian chu kì ghi dữ liệu tối đa 5ms đối với 24c04
- Thời gian cần giữ ở trạng thái START/STOP 0.6µs
- Thời gian thiết lập data (SDA) trước khi đưa SCL lên cao tối thiểu 0.1us
- Thời gian để có data out tối đa 0.9us
  ⇨ phải chờ ít nhất 1us sau khi SCL xuống thấp mới bắt đầu lấy dữ liệu ở ngõ SDA

Chức năng của các chân:

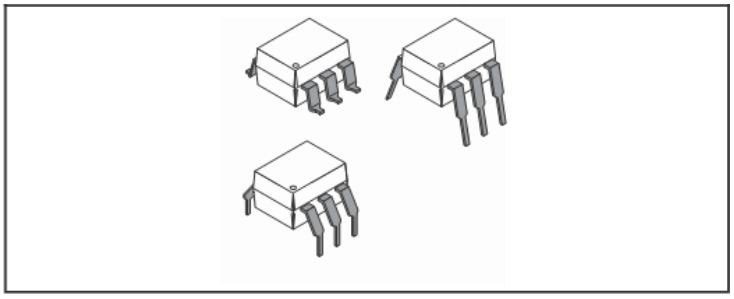
- SDA: là chân dữ liệu nối tiếp, có thể truyền địa chỉ và dữ liệu 2 chiều. Đây là chân open drain cho nên cần mắc thêm điện trở pull -off khoảng 1k đối với tốc độ truyền 100kHz và 2k đối với tốc độ 400kHz. 1
  1 số chú ý đối với SDA:

  + Khi chuyển dữ liệu SDA chỉ được thay đổi khi SCL xuống thấp.

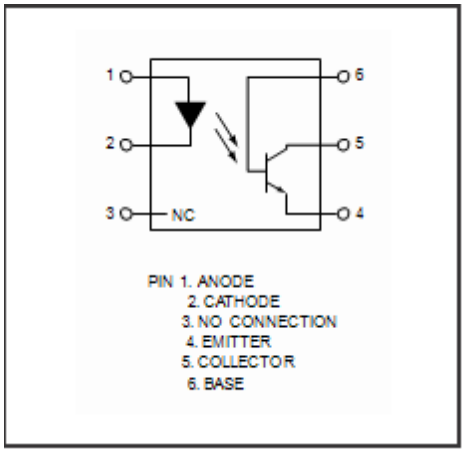
+ SDA thay đổi khi SCL ở cao dùng để báo hiệu điều kiện bắt đầu (START) và kết thúc (STOP)

- SCL: xung clock dùng để đồng bộ hóa việc truyền dữ liệu
- ngõ vào SCL và SDA đã có mạch lọc để triệt tiêu nhiễu

3. ***4n35***



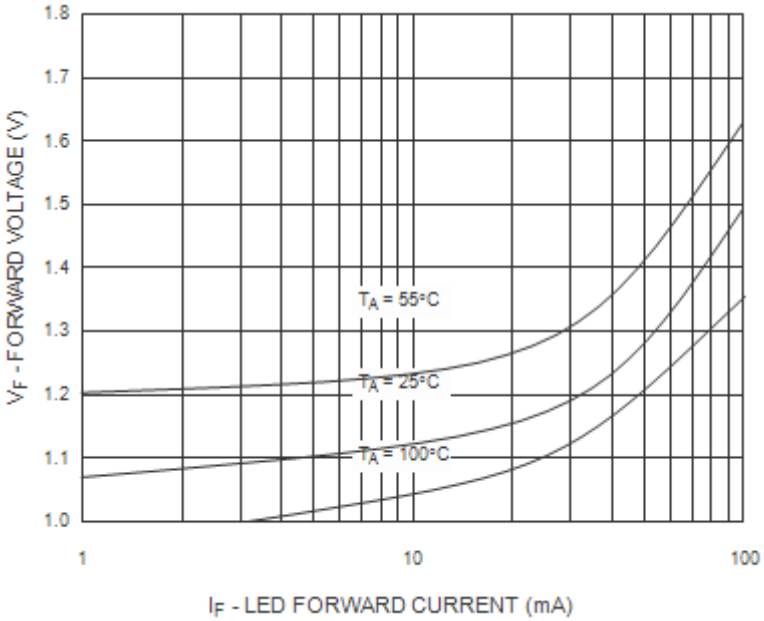
Hình 6-Dạng bên ngoài của 4n35



Hình 5-Cấu tạo chân của 4n35

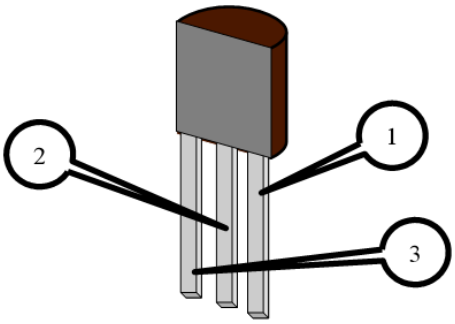
Cấu tạo của 4n35 gồm 1 photoDiode và 1 phtotransistor. Khi có dòng qua photoDiode, ánh sáng phát ra từ photoDiode sẽ làm photoTransistor dẫn.

- Ứng dụng của 4n35 dùng để tạo tín hiệu cho vi xử lí.
- Đặc tính dòng và áp của photoDiode trong 4n35 như sau:



Hình 7 - Đặc tính dòng, áp của 4n35

4. ***Um66***



Hình 8 - Sơ đồ chân UM66

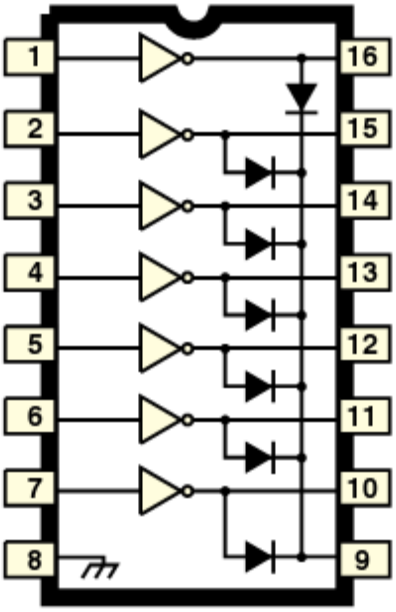
Đây là 1 IC nhạc đơn giản, giá rẻ, nó được nạp sẵn 1 bài nhạc đơn âm. Ta có thể tận dụng nó để tạo âm thanh phản hồi cho người dùng.

Sơ đồ chân của UM66 như sau:

1	2	3
Out	Vin	GND

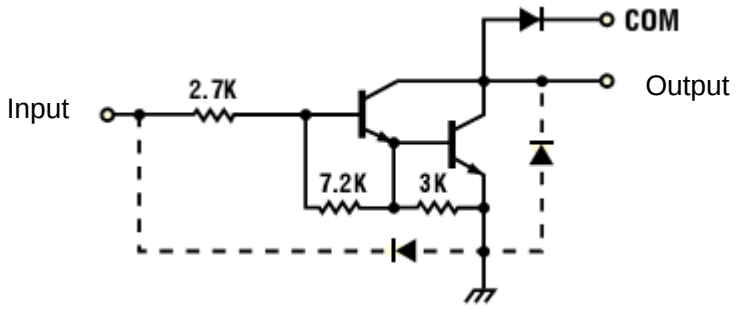
Thông thường Vin khoảng 3V, điện áp ở chân Out khoảng 0.8V.

5. ***ULN2003:***



Hình 9 - Sơ đồ khối ULN2003

Bên trong ULN2003 là một loạt các cặp transitor NPN mắc darlington theo sơ đồ sau:



Hình 10 - Cấu tạo một cặp Darlington

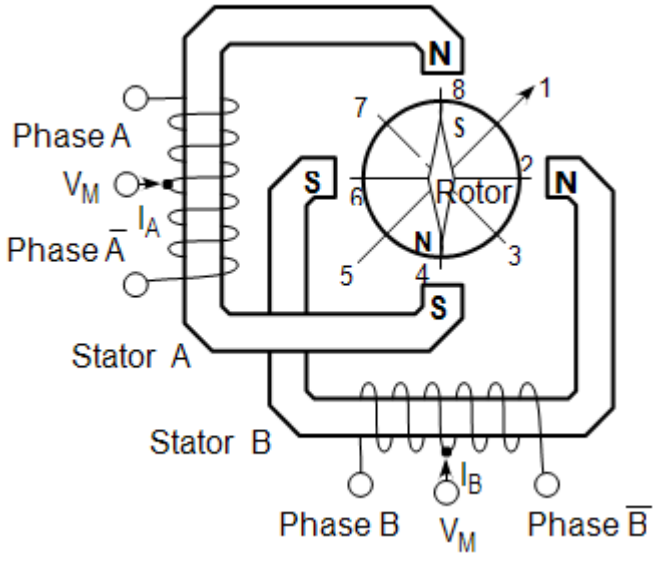
Khi đầu vào Input ở mức cao , cặp darlington dẫn cho phép dòng đổ từ chân output xuống mass . Điện áp ở cổng COM có tác dụng giới hạn áp ở ngõ Output , $V_{Output}>V_{COM}+V_D$ thì toàn bộ dòng đổ ra ngoài qua cổng COM.

Ngược lại, khi Input ở mức thấp , cặp trasistor khóa lại , không cho dòng đổ vào từ Output.

Xét trên phương diện logic thì có thể xem Output là cổng đảo của Input . Tuy nhiên, mức cao của output không nhất thiết là 5V mà có thể là 12V hay bao nhiêu tùy nguồn ngoài mắc vào output.

Ta sẽ dùng ULN2003 làm bộ đệm điều khiển cho động cơ.

6. ***Động cơ bước:***



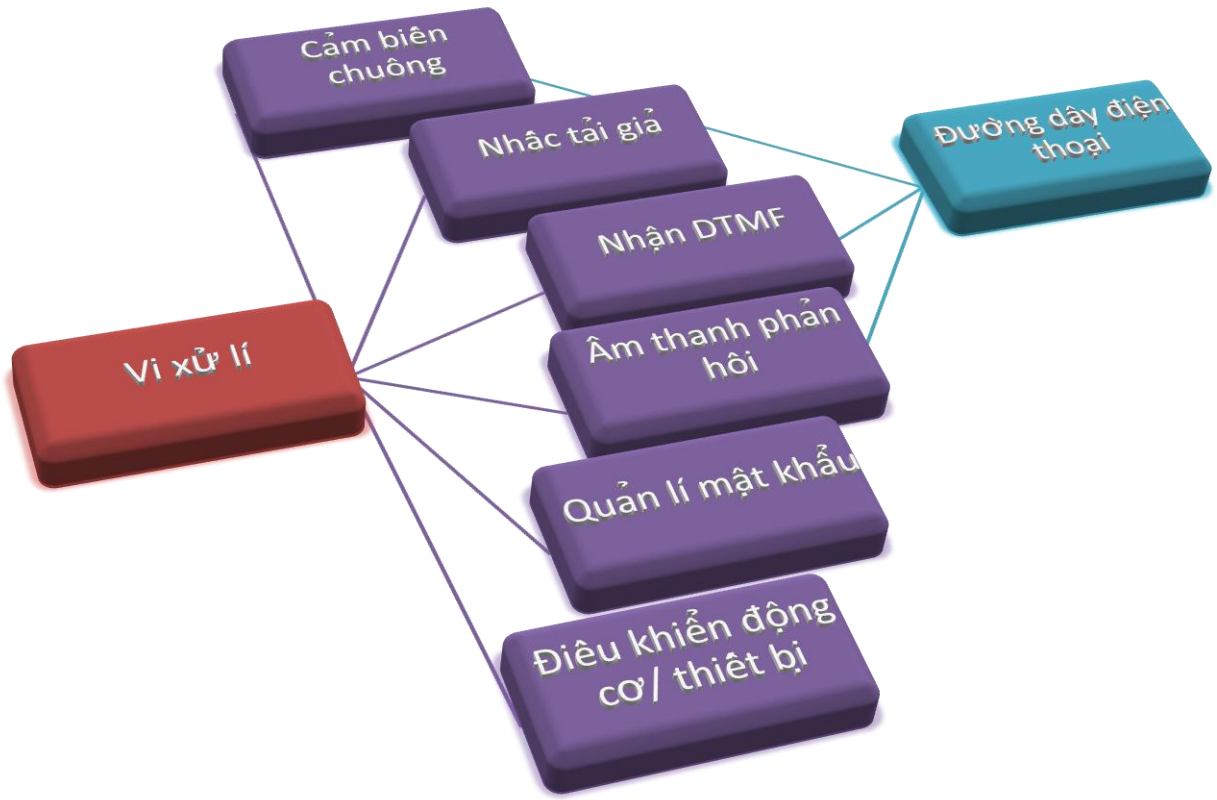
Hình 11 - Cấu tạo động cơ bước

Động cơ bước loại 6 dây có cấu tạo như hình trên. Phần rotor có 1 nam châm vĩnh cửu còn phần stator gồm 2 nam châm điện được gắn cố định . Các cực của nam châm điện stator do chúng ta điều khiển, bằng cách cấp các giá trị áp ở các chân A, $\overline{A}$, B, $\overline{B}$, $V_{M.}$

Một cách đơn giản để điều khiển động cơ bước loại này là cấp sẵn điện áp ở 2 chân V_M=12V, sau đó cho các chân A, $\overline{A}$, B, $\overline{B}$ xuống thấp, biến stator tương ứng thành các nam châm với các cực khác nhau, hút rotor quay, làm động cơ quay.

III. THỰC HIỆN PHẦN CỨNG

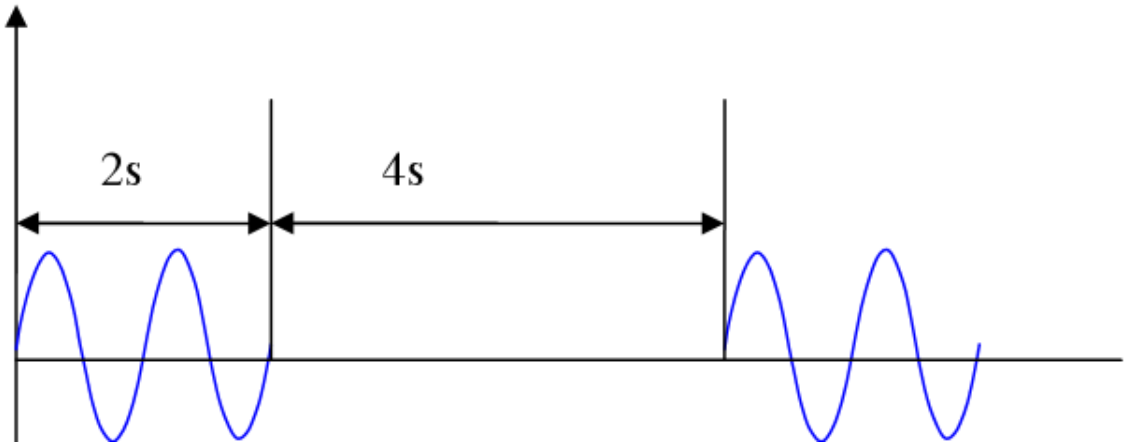
1. Sơ đồ khối:



Hình 12 – Sơ đồ khối phần cứng

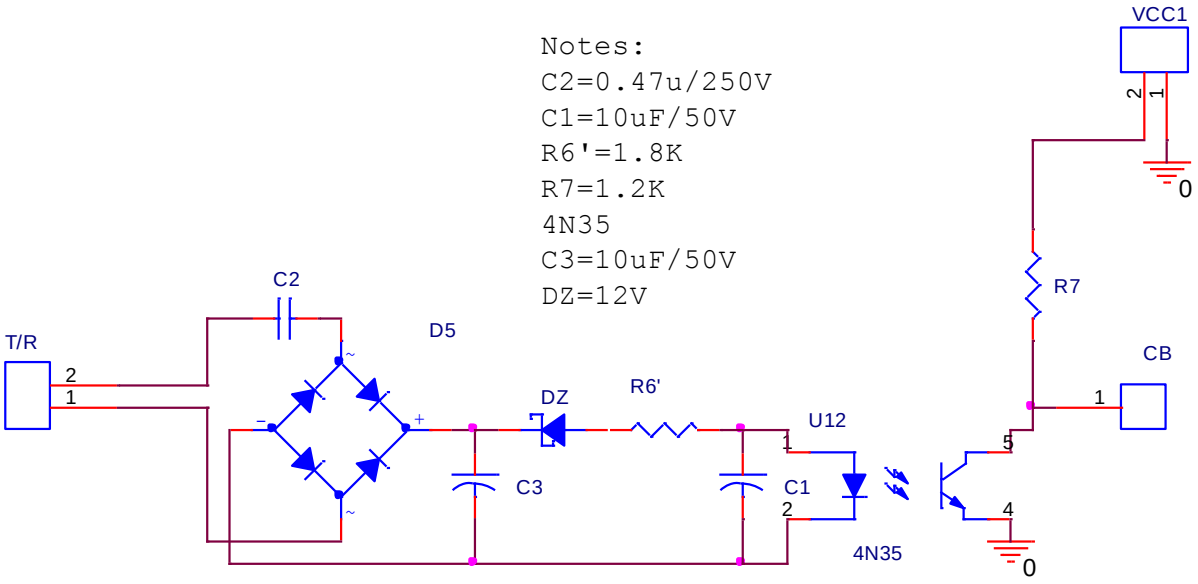
2. ***Mạch nhận tín hiệu chuông:***

Tín hiệu chuông có dạng từng chuỗi hình sine ngắt quãng, 2s có, 4s không.



Hình 13- Tín hiệu chuông

Biên độ 75→90 Vrms, tần số 25Hz



Hình 14 - Mạch nhận tín hiệu chuông

C2 có tác dụng ngăn dòng DC, đồng thời giảm biên độ tín hiệu chuông vào mạch cảm biến.

Diode cầu dùng để chỉnh lưu tín hiệu vào, tăng gấp đôi tần số.

C3, C1: lọc để tín hiệu ổn định, ít nhấp nhô. Tấn số tín hiệu qua diode cầu được tăng gấp đôi nên lọc dễ dàng hơn

Dz: lọc nhiễu , tín hiệu < 12V không thể đi qua.

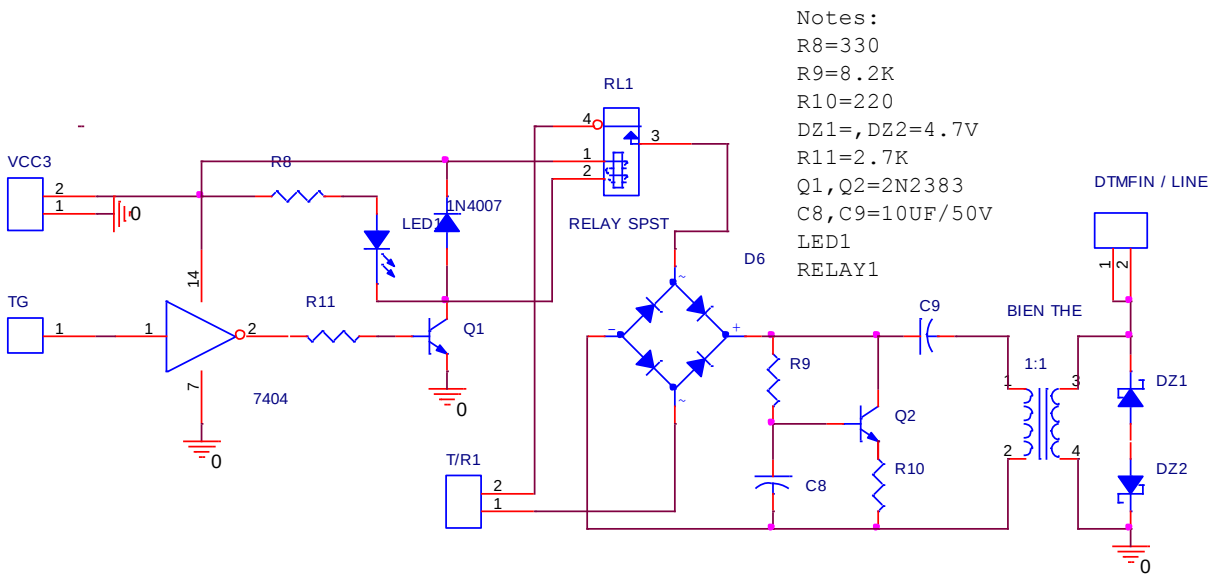
Opto 4N35: khi có chuông, photodiode bức xạ làm phototransistor dẫn, ngõ ra của mạch cảm biến ở mức 0. Ngoài ra, opto còn dùng để tách tầng cảm biến chuông và tầng xử lí tín hiệu.

Tóm lại: ngõ ra của mạch cảm biến chuông ở mức

+ 0 khi có tín hiệu chuông

+ 1 khi không có tín hiệu chuông

3. ***Mạch đóng tải giả:***



Hình 15 - Mạch đóng tải giả

Mạch đóng tải giả gồm 2 phần:

+ Phần đóng ngắt relay

+ Phần tạo tải giả

Khi có tín hiệu điều khiển từ vi xử lí qua ngõ TG thì Q1 dẫn, tạo dòng qua relay, làm đóng phần mạch tạo tải giả

Diode cầu ở phần mạch tạo tải giả có tác dụng chống đảo cực (tín hiệu đảo cực thường dùng để tính cước điện thoại).

Tổng đài nối với các thuê bao thông qua 2 dây TIP và RING. Dòng đi qua đường dây này có giá trị trong khoảng 25 mA đến 40 mA (trung bình chọn 35 mA)

Tổng trở DC khi gác máy luôn >20 K Ω

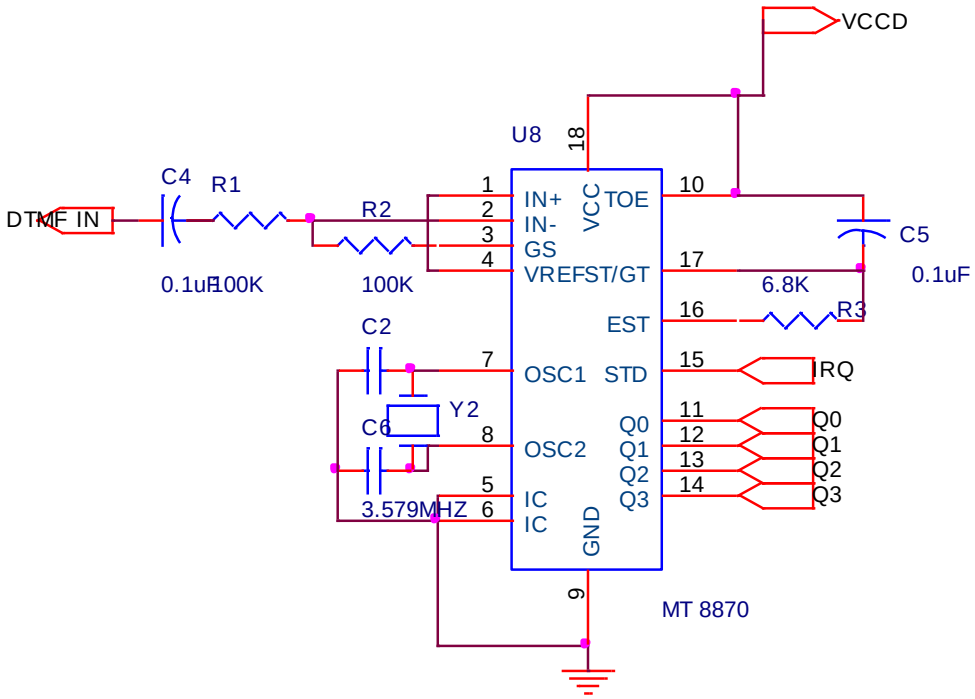
Tổng trở AC khi gác máy từ 4KΩ đến 10K Ω

Tổng trở DC khi nhấc máy <1K (từ 0,2K đến 0,6K).

Do đó, chức năng của phần tạo tải giả đó là thay thế 1 thuê bao thật sự về mặt trở kháng. Trở kháng DC của tải giả sẽ <300Ω, còn trở kháng AC của nó trong khoảng 700Ω±30%.

Tụ C3 sẽ ngăn tín hiệu DC , chỉ cho t ín hiệu AC mang nội dung thoại hay DTMF đi qua. Ở đây, chúng ta dùng 1 biến thế 1:1 để cách li tín hiệu AC giữa phần tín hiệu vào với phần xử lí tín hiệu và phản hồi (DTMF IN – lấy tín hiệu DTMF từ bên ngoài vào/LINE – tín hiệu phản hồi)

4. ***Mạch nhận và giải mã tín hiệu DTMF:***

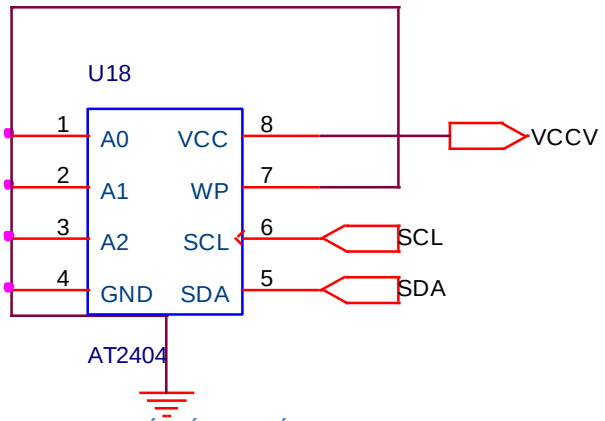


Hình 16 - Cách mắc 8870

Nếu phát hiện cặp tone DTMF thì 8870 sinh ra 1 ngắt ở chân STD, đồng thời 4**bit** mã tương ứng với tín hiệu DTMF nhận được đưa đến các chân Q 0,Q1,Q2,Q3. Các chân này được đưa đến vi xử lí.

5. *Mạch truy xuất bộ nhớ chứa mật mã:*

24C04 là bộ nhớ nối tiếp, sử dụng 2 chân SCL và SDA để giao tiếp với vi xử lí trung tâm. Ta nối 2 chân này đến vi xử lí , các chân còn lại (trừ chân nguồn) nối đất. Như

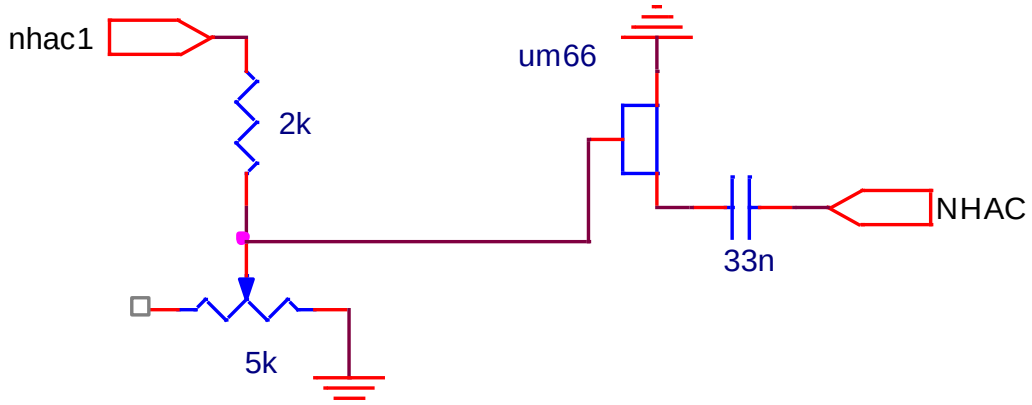


Hình 17 - Kết nối bộ nhớ

vậy, địa chỉ của 24C04 khi giao tiếp với vi xử lí là 1010000.

6. *Mạch tạo tín hiệu phản hồi:*

Để phản hồi các thao tác của người dùng , ta dùng chip UM66 để tạo tiếng nhạc. Chân 'nhac1' lấy nguồn từ vi xử lí, chân 'NHAC' đưa vào đường LINE sau biến thế 1:1 của



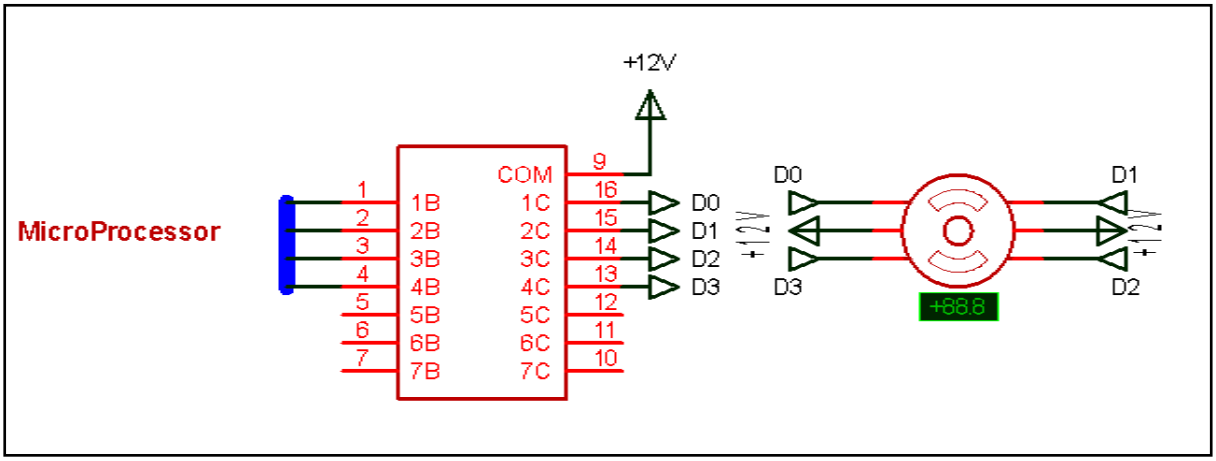
Hình 18 - Tạo tín hiệu phản hồi

mạch đóng tải giả.

Biến trở 5k có tác dụng chỉnh âm lượng của tiếng nhạc.

Ngoài UM66, ta có thể dùng loại chip nhạc của Trung Quốc , giá 1000đ/1 con ở chợ Nhật Tảo.

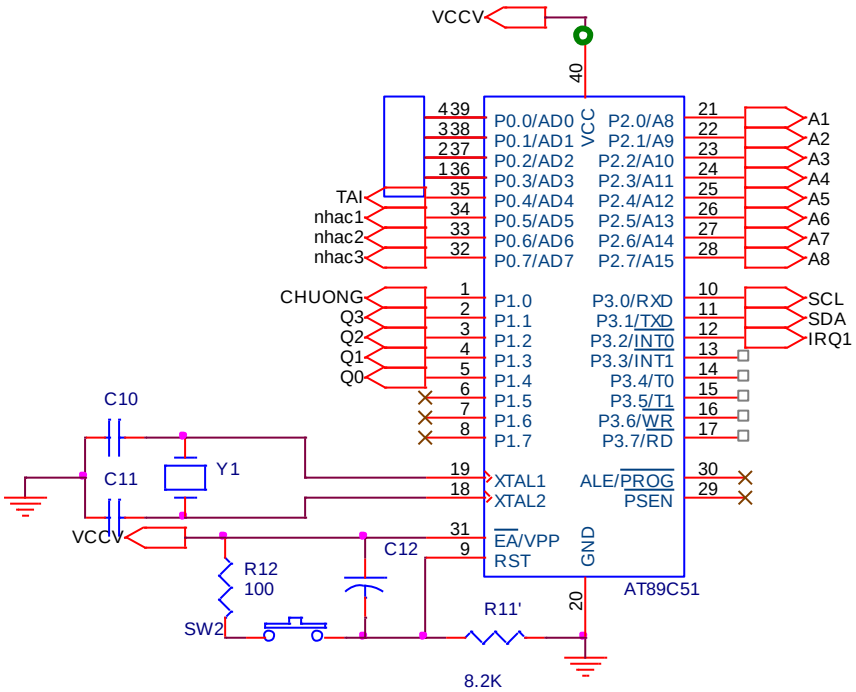
7. *Mạch điều khiển động cơ bước:*



Hình 19 - Điều khiển động cơ bước thông qua ULN2003

Loại động cơ bước được sử dụng trong đề tài này không hoạt động nếu cấp mức điện áp 5V, vì vậy, cần phải cấp nguồn ngoài 12V cho nó, đồng thời dùng bộ đệm ULN2003 để giao tiếp với vi xử lí. Lúc này mức logic cao của D 0, D1, D2, D3 tương ứng với 12V. Mức logic của các chân iC là nghịch đảo của các chân iB.

8. *Vi xử lí trung tâm:*



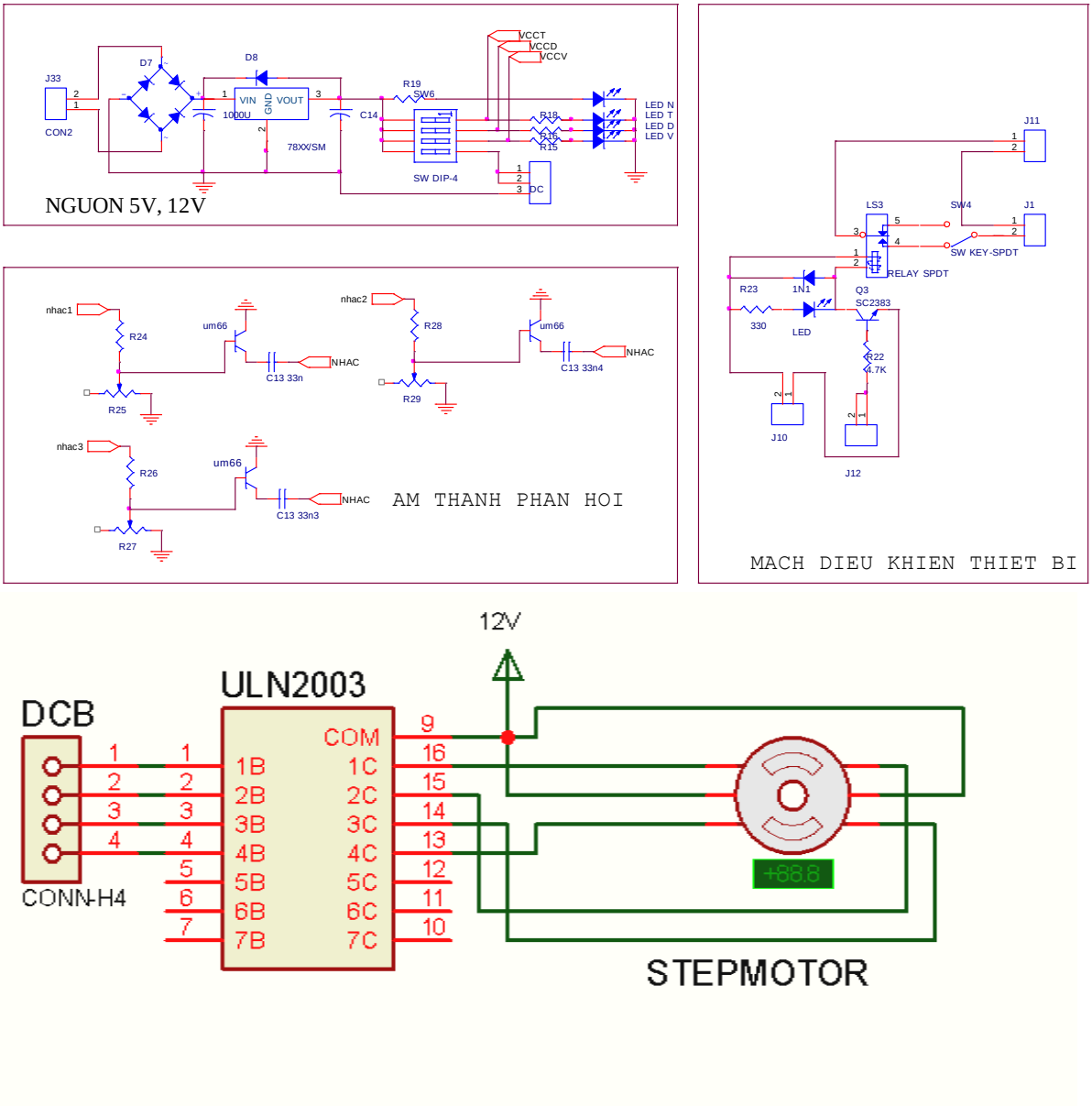
Hình 20 - Vi xử lí trung tâm

Ta thiết lập cách kết nối như trên hình, trong đó:

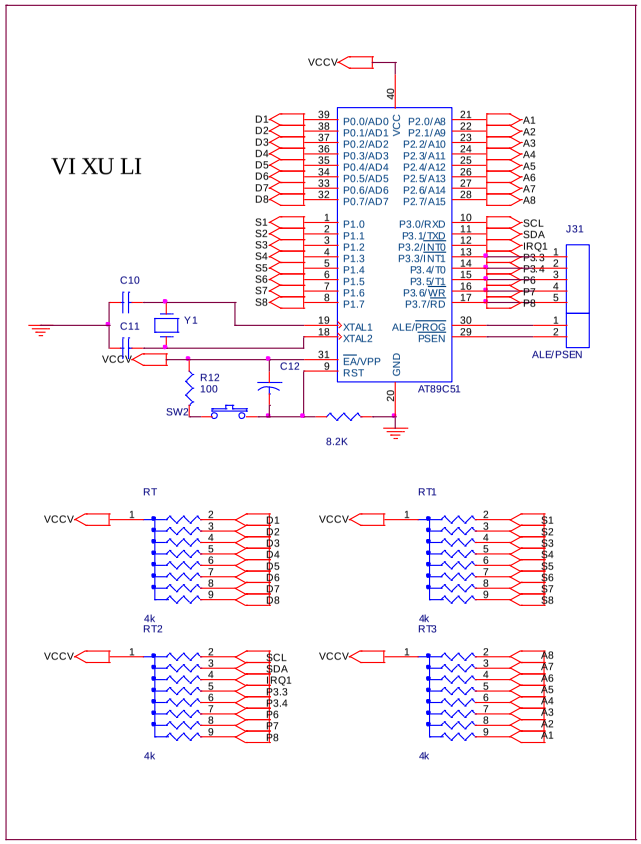
- Chân P0.0 → P0.3 :điều khiển động cơ bước
- Chân P0.4 :nhắc tải giả, tích cực ở mức thấp

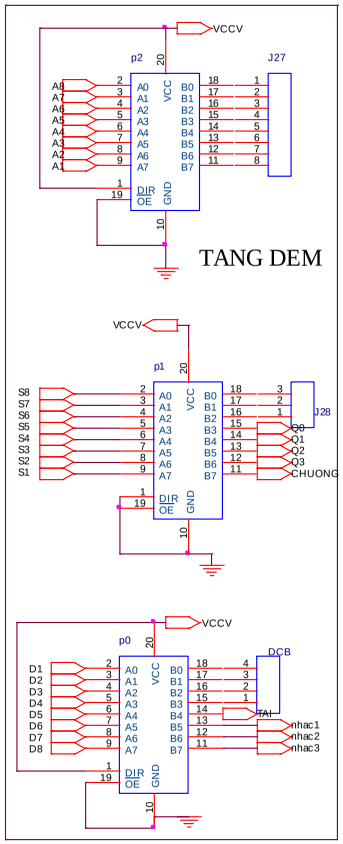
- Chân P0.5 → P0.7 :phát tiếng nhạc, phát tín hiệu phản hồi
- Chân P1.0 :cảm biến chuông (xuống 0 khi có chuông)
- Chân P1.1 → P1.4 :mã tương ứng với tín hiệu DTMF nhận được (MSB:P1.1;LSB:P1.4)
- Chân P3.0 :nối với SCL của 24C04
- Chân P3.1 :nối với SDA của 24C04
- Chân P3.2 :nhận ngắt từ 8870

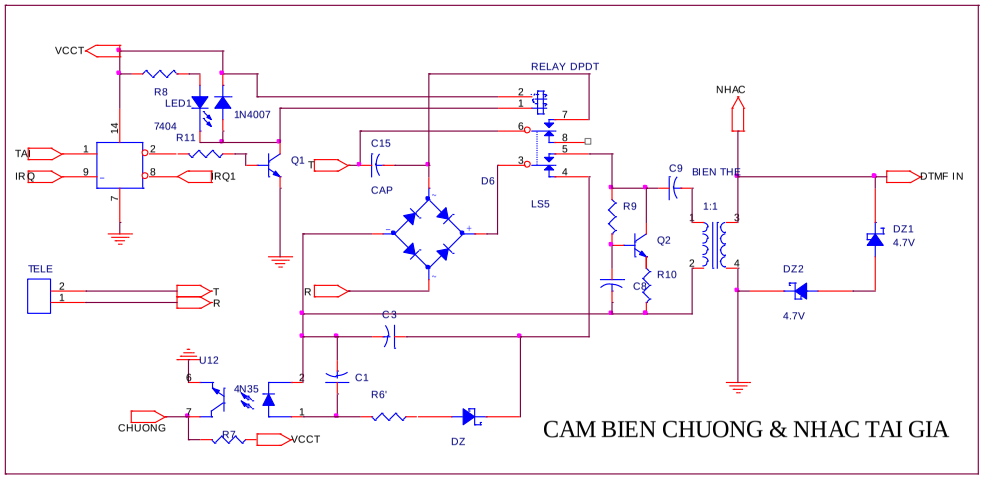
9. ***Sơ đồ mạch:***

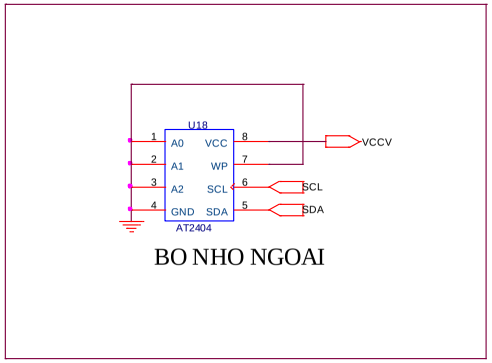


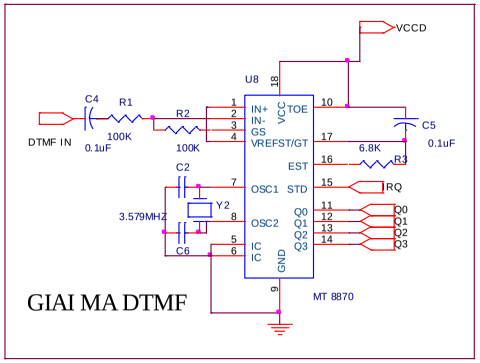
Hình 21(a) - Sơ đồ mạch











Hình 22(b) - Sơ đồ mạch

VI XU LI:
89C51
CRYSTAL 12MHz
C10,C11=33PF
C12=0.1uF
R11'=8.2K
RT1-4=4.7k
R12=100

CAM BIEN CHUONG & NHAC TAI GIA

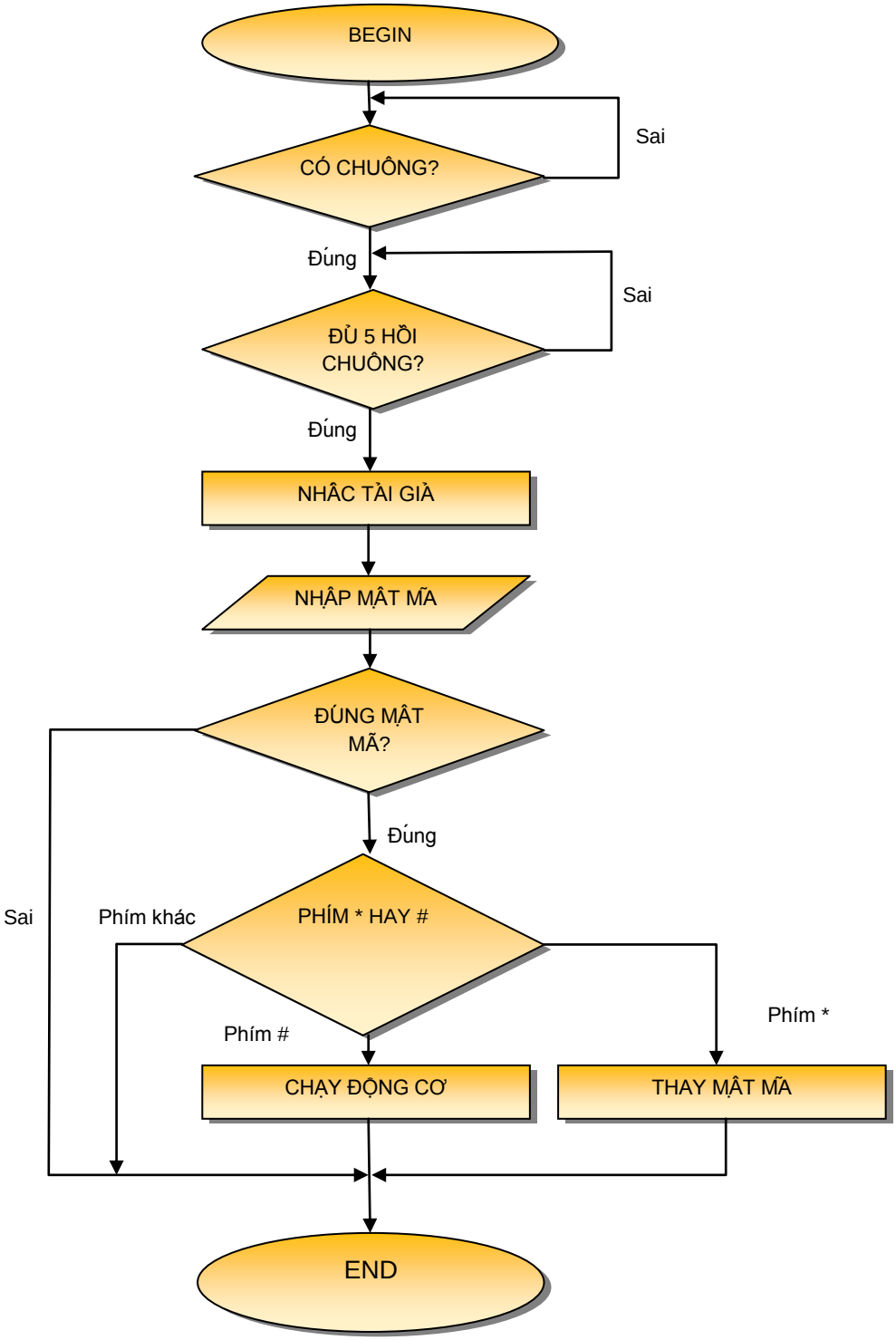
R6'=1.8K	Q1,Q2=2N2383
R7=1.2K	C1=10uF/50V
R8=330	C3=10uF/50V
R9=8.2K	C8,C9=10UF/50V
R10=220	C15=0.47u/250V
R11=2.7K	DZ1=DZ2=4.7V
R16=3k	DZ=12V
	4N35

GIAI MA DTMF:
R1, R2 = 100 k 1%
R3 = 6.8 k 1%
C4= 100 nF 5%
C5= 100 nF 10%*
C6 =C2 33pF 10%
X-tal = 3.579545 MHz

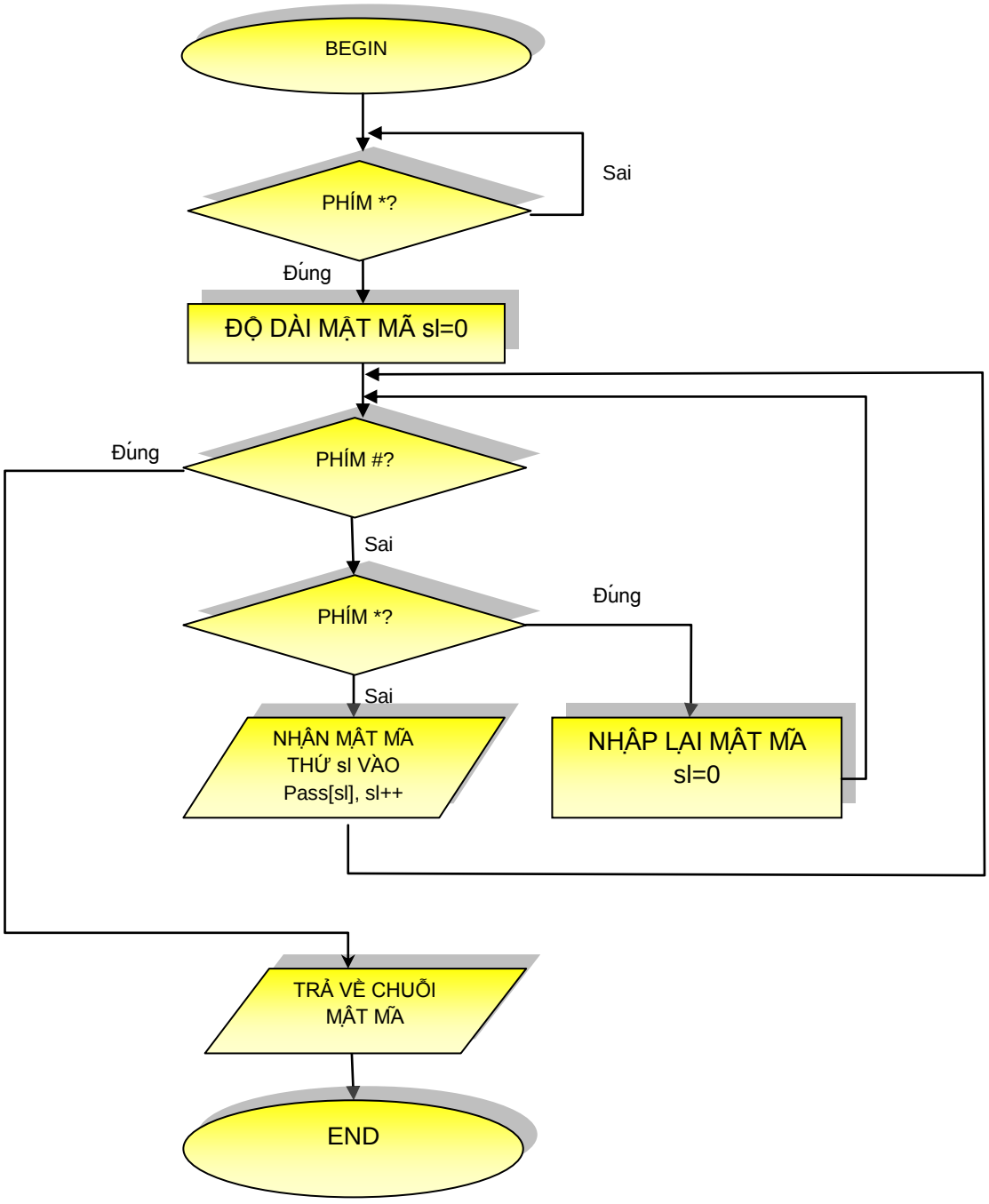
-

IV. Lập trình:

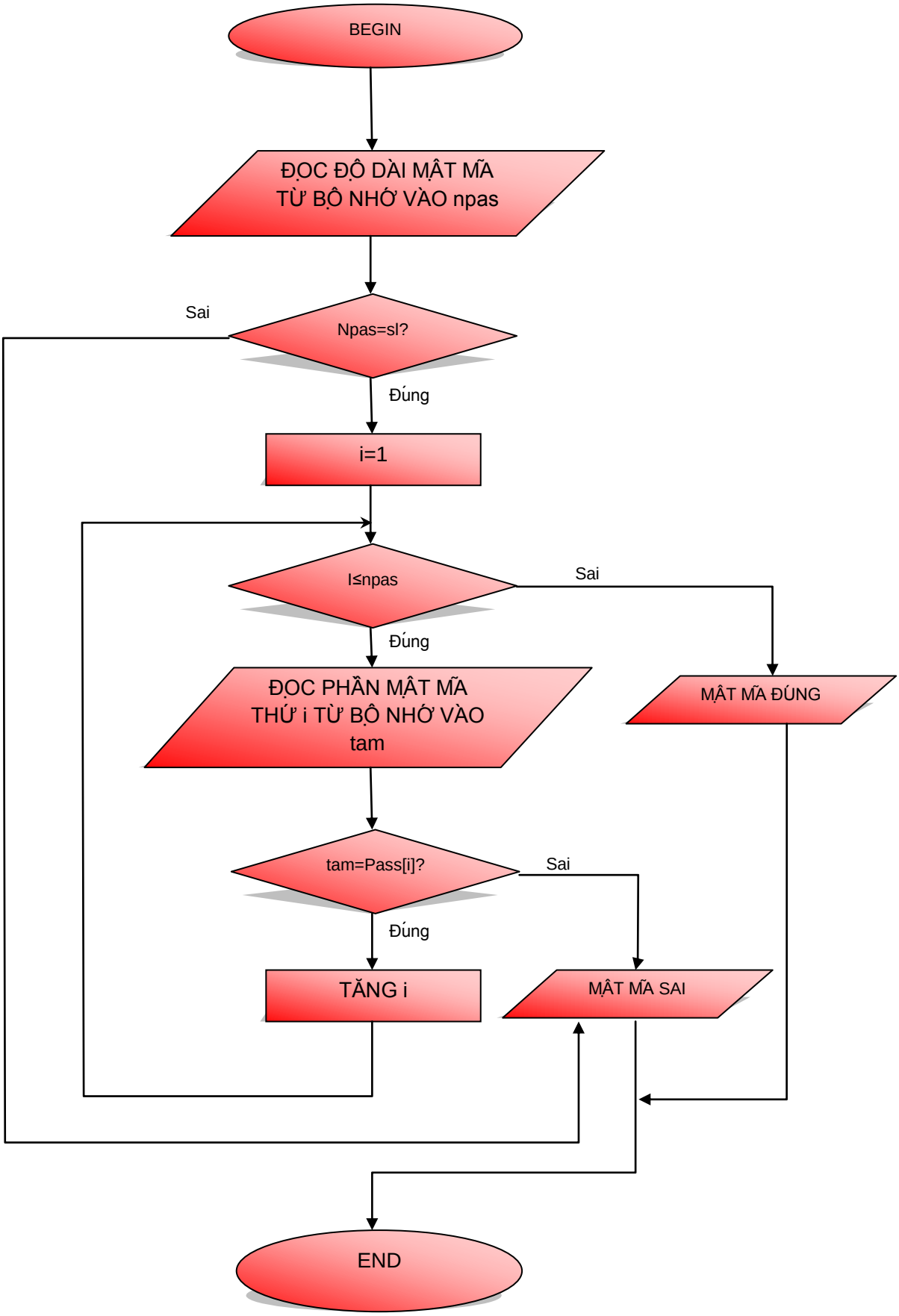
1. *Phần chính của chương trình*



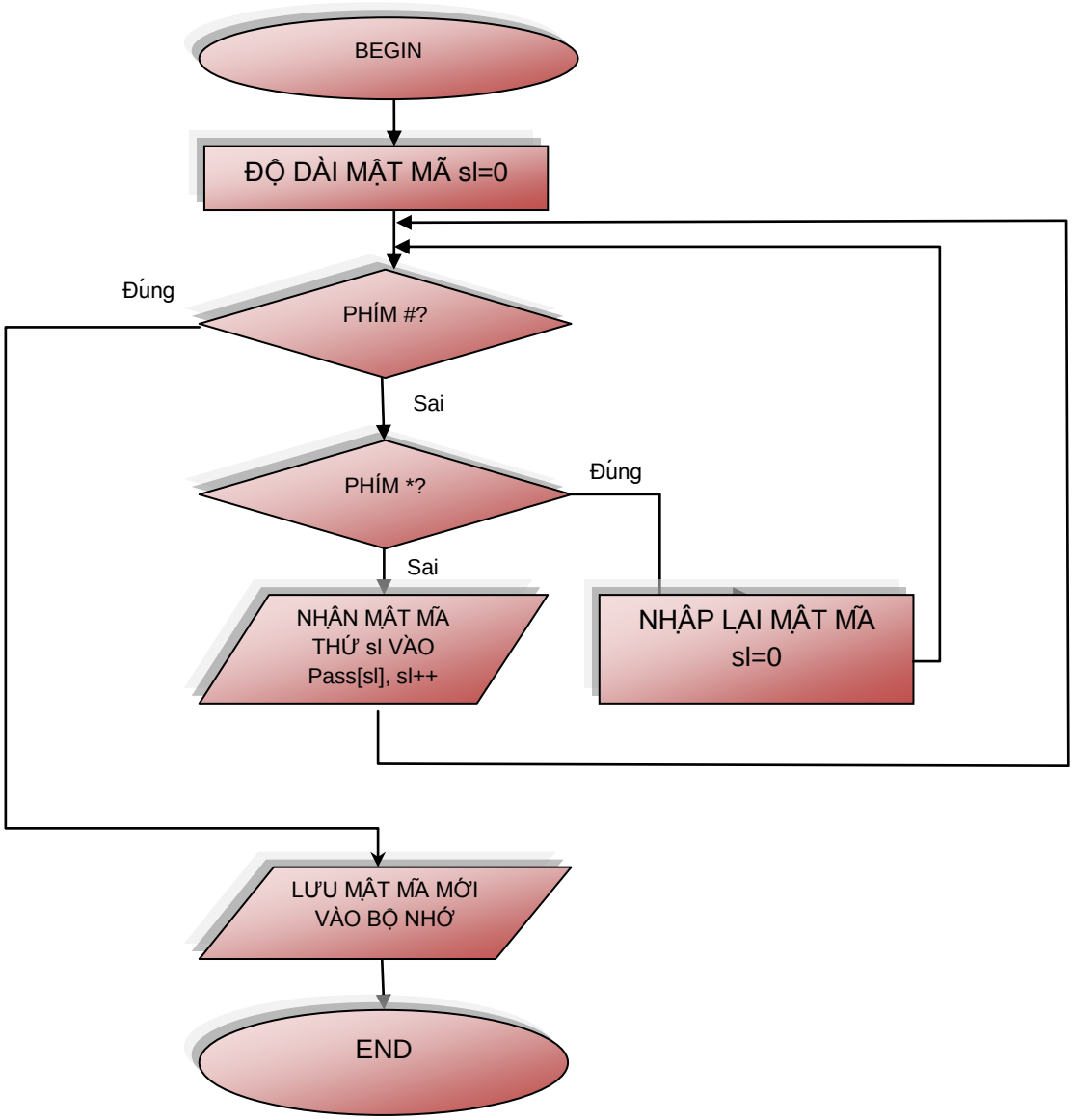
2. ***Phần nhận mật mã:***



3. *Phần kiểm tra mật mã:*

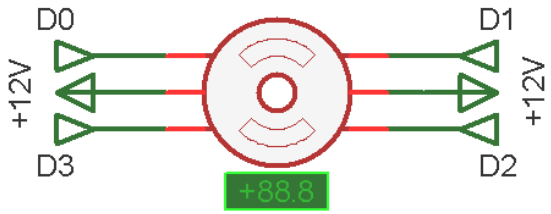


4. *Phần thay mật mã:*



5. ***Phần điều khiển động cơ bước:***

Có 3 chế độ hoạt động cho động cơ bước loại 6 dây sử dụng trong đề tài. Muốn động



Hình 23 - kết nối với động cơ

cơ hoạt động trong chế nào thì ta cấp các **bit** tương ứng với các chân D0, D1, D2, D3 như bảng sau:

Phase	Wave Drive 1 2 3 4	Normal full step 1 2 3 4	Half-step drive 1 2 3 4 5 6 7 8
A =D0	•	• •	• •
B =D1	•	• •	• • •
$\overline{A}$ =D2	•	• •	• • •
$\overline{B}$ =D3	•	• •	• • •

6. ***Phần truy xuất bộ nhớ:***

Phần lập trình cho 24c04 được đánh giá là khó nhất trong toàn bộ chương trình . Phải nắm kĩ hoạt động của 24c04 thì mới làm được.

Các trạng thái trên đường BUS 2 dây SDA và SCL:

- BUS rảnh

  + SDA và SCL đều lên cao

  + Trạng thái này phải được duy trì ít nhất 4.7µs giữa mỗi lần đọc, ghi dữ liệu

- Bắt đầu truyền dữ liệu

  + SDA từ 1 → 0 khi SCL==1 : START

- Kết thúc truyền dữ liệu

  + SDA từ 0 → 1 khi SCL==1: STOP

- Nhận dữ liệu

  + Sau START, SDA cố định khi SCL=1 → nhận dữ liệu

  + Dữ liệu chỉ được thay đổi khi SCL=0

  + Trong mỗi chu kì xung, chỉ có tối đa 1 **bit** được truyền

- Xác nhận (ACK):

  + Giữa các byte dữ liệu phải bổ sung 1 clock để xác nhận byte

  + Để xác nhận byte: phải kéo SDA line xuống thấp (giữ SDA==0 suốt quá trình SCL==1). Việc này do 24c04 đảm nhận trong quá trình ghi dữ liệu.

  + Kết thúc byte cuối cùng thì không xác nhận mà cần gi ải phóng BUS (SDA==1 khi SCL==1)

Khởi động trước khi dùng 24c04

Sau khi reset hệ thống hay sau khi STOP, 24C04 cần được reset theo các bước

+ Trong nhiều nhất 9 chu kì clock

+ Xét xem SDA có ở mức cao hay không mỗi khi SCL ở mức cao

+ Nếu có tạo tín hiệu START

Đặc tính BUS:

+ Điều kiện truyền dữ liệu: BUS không bận

+ Đường data phải ổn định khi clock lên cao

+ Thay đổi dữ liệu trong lúc clock lên cao ứng với đk START và STOP

Đánh địa chỉ

+ Sau START, µP phải chuyển 1 byte điều khiển chứa địa chỉ của 24LC04 và 1 **bit** R/~W (cho phép đọc hay ghi)

+ Byte địa chỉ của 24C04 có dạng 1010xxxY với Y là R/~W

+ Khi bắt đầu truyền dữ liệu, dạng **bit** chuyển qua lại như sau: S1010xxxYZ với

- S: START

- Y: R/~W

- Z: ACK

+ 24C04 giám sát BUS liên tục để xác nhận địa chỉ tương ứng của nó

+ 24C04 phát sinh **bit** acknowledge nếu đúng địa chỉ và nó không ở trong chế độ bận (đang ở trong quá trình ghi nội chẳng hạn)

GHI BYTE:

+ Sau loạt tín hiệu S1010xxxYZ (Z do 24C04 gây ra) là byte địa chỉ của dữ liệu cần ghi

+ Byte địa chỉ này được ghi vào con trỏ địa chỉ của 24C04

+ 24C04 xác nhận byte địa chỉ.

+ µP chuyển byte dữ liệu cho địa chỉ đã xác định trong 24C04

+ 24C04 xác nhận lần nữa

+ µP tạo tín hiệu STOP

+ Đợi 5ms cho quá trình ghi (tiến hành trong 24C04)

- *Chú ý*

+ Bộ đếm địa chỉ nội sẽ không tăng lên mà giữ địa chỉ cũ sau khi ghi xong

+ Nếu có tín hiệu STOP khi chưa kết thúc dãy lệnh ghi thì quá trình ghi sẽ được hủy

+ Nếu nhiều hơn 8 **bit** dữ liệu được gửi trước lệnh STOP thì 24C04 xóa các byte đã load trước đó và load lại buffer

+ Nếu có nhiều byte được chuyển trong 1 lệnh thì chỉ có byte cuối được lưu.

+ Nếu có nhiều byte được chuyển trong 1 lệnh thì trong quá trình truyền , nếu byte cuối chưa đủ 8 **bit** thì quá trình ghi sẽ bị hủy

+ 24C04 dùng mạch xác định ngưỡng Vcc để tắt chế độ ghi/xóa logic nếu Vcc<1.5V (24AA00 và 24C04) hay <3.8V đối với 24C00

XÁC NHẬN (acknowledge polling)

+ 24C04 không xác nhận chừng nào chưa ghi xong → nhận biết khi nào chu kì ghi kết thúc

+ Khi µP ra lệnh STOP cuối dãy lệnh ghi dữ liệu, 24C04 bắt đầu chu kì ghi nội.

+ Khi gửi dãy lệnh ghi S1010xxxYZ, nếu 24C04 đang bận thì không có tín hiệu xác nhận, phải gửi lại dãy lệnh ghi

ĐỌC

+ Thiết lập như quá trình ghi nhưng Y==1

+ Có 3 kiểu đọc dữ liệu cơ bản

- Đọc tại địa chỉ hiện tại

- Đọc ngẫu nhiên

- Đọc theo thứ tự

(Ghi chỉ có 1 kiểu: ghi tại địa chỉ xác định)

- Đọc tại địa chỉ hiện tại

+ Bộ đếm dữ liệu trỏ đến địa chỉ byte sau cùng được truy xuất

+ Tự động tăng lên 1 sau mỗi lần đọc

+ Sau dãy lệnh đọc, 24C04 tạo tín hiệu ACK và truyền 8 **bit** dữ liệu

+ µP không xác nhận việc truyền dữ liệu mà sinh tín hi ệu STOP và 24C04 ngừng gửi dữ liệu

- Đọc ngẫu nhiên

+ Phải có 1 địa chỉ dữ liệu thiết lập trước

+ Thiết lập bằng địa chỉ trong lệnh ghi

+ Sau khi địa chỉ dữ liệu được gửi thì µP gửi tiếp START làm quá trình ghi kết thúc nhưng con trỏ địa chỉ vẫn nằm ở vị trí được thiết lập trong dãy lệnh ghi trước đó.

⇨ Dãy lệnh gửi đi có dạng S1010xxxYZxxxxxxxxZS (Y=0: ghi, Z: do 24C04)

+ Gửi tiếp dãy lệnh điều khiển với Y=1 S1010xxxY

+ 24C04 gửi ACK và dãy 8 **bit** dữ liệu

+ µP phát sinh tín hiệu STOP

+ Bộ đếm địa chỉ sẽ trỏ đến địa chỉ tiếp theo sau địa chỉ vừa đọc

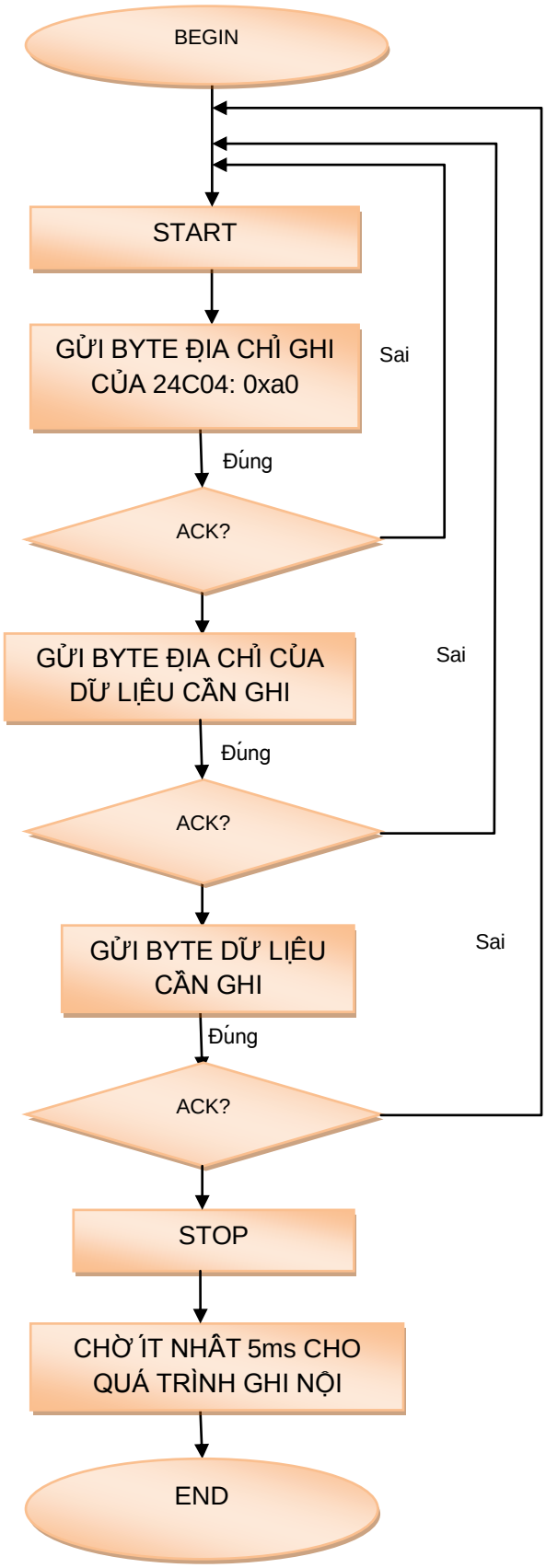
- Đọc theo thứ tự

+ Thiết lập ban đầu giống như đọc ngẫu nhiên

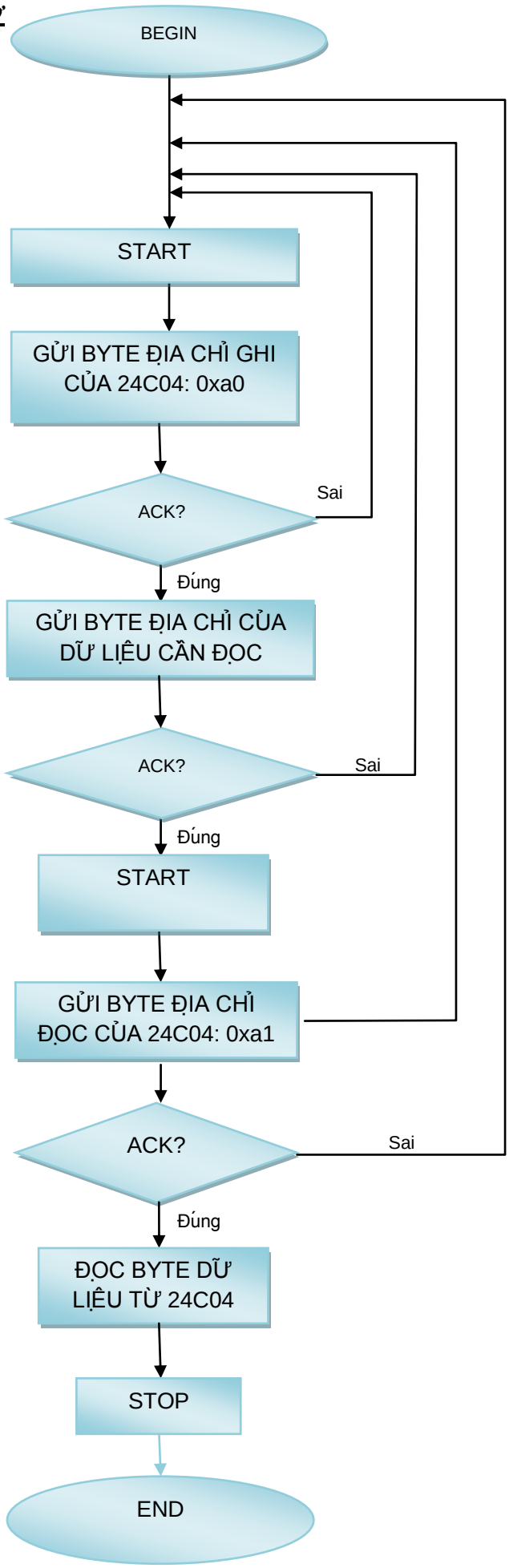
+ Sau khi đọc byte đầu tiên, không phát sinh STOP mà tạo ACK

+ 24C04 sẽ tiếp tục gửi tiếp 8 **bit** dữ liệu kế

a. Phần ghi bộ nhớ:



b. *Phần đọc bộ nhớ*



7. *Đoạn code chính của chương trình (viết bằng C)*

```
#include <reg51.h>
#include <intrins.h>
#define NGAT0 0
#define NGAT1 3
      //timer1
#define max 40

typedef unsigned char ii;
sbit kochuong=P1^0;
sbit taigia=P0^4;

sbit SDA=P3^1;
sbit SCL=P3^0;
sbit kongat=P3^2;

sbit B3=P1^1;
sbit B2=P1^2;
sbit B1=P1^3;
sbit B0=P1^4;

sbit nhac1=P0^5;
sbit nhac2=P0^6;
sbit nhac3=P0^7;

ii pass[max];
ii npas,sl;
bit started,xong;
ii buoc;
ii nwait;

void delay(int t)
{
int j;
for (j=0;j<1000*t;j++)
_nop_();
}

void delay_i2c()
{
while(0);
while(0);
}

void start_i2c()
{
SDA=1;
SCL=1;
delay_i2c();
SDA=0;
delay_i2c();
SCL=0;
delay_i2c();
}
void stop_i2c()
{
SDA=0;
SCL=0;
delay_i2c();
SCL=1;
delay_i2c();
SDA=1;
delay_i2c();
}
ii rx_i2c(ii ACK)
{
ii d=0,x;
SDA=1;// tha noi cong
// qua trinh doc du lieu tu
ngoai vao
for (x=0;x<8;x++)
{
d<<=1;
SCL=1;
while(SCL==0);
// khi da co canh len thi doc
du lieu tu ngoai vao
delay_i2c();
if (SDA) d|=1;
SCL=0;
}
//
// tao xung ACK luu y la co
xet xem la co can tao xung ACK
khong
if (ACK) SDA=0;
else SDA=1;
SCL=1;
delay_i2c();
SCL=0;
SDA=1;//release the SDA line
return d;
}
bit tx_i2c(ii trans)
{
ii x;
bit ACK;
SCL=0;
for (x=0;x<8;x++)
{
if(trans&0x80) SDA=1;
else SDA=0;
trans<<=1;
```

```
SCL=1;
delay_i2c();
SCL=0;
delay_i2c();
}
//giai phong duong truyen de
doi tin hieu ack
SDA=1;
SCL=1;
//
delay_i2c();
while(SCL==0);
ACK=SDA;
SCL=0;//pull down all line
return ACK;
}

void write(ii dc,ii dl)
{
bit ok=0;
while (!ok)
{
start_i2c();
if (tx_i2c(0xA0)) {continue;}
if (tx_i2c(dc)) {continue;}
if (tx_i2c(dl)) {continue;}
stop_i2c();
delay(3);//chu ki ghi du lieu
ok=1;
}
}


ii read(ii dc)
{
ii d;
while (1)
{
start_i2c();
//tao dia chi de doc
if (tx_i2c(0xA0)) {continue;}
if (tx_i2c(dc)) {continue;}
start_i2c();
if(tx_i2c(0xA1))
{continue;}//read
d=rx_i2c(0);//doc 1 byte voi
xung N_ACK
stop_i2c();
return d;
}
}

void cho(int t)          //cho
ts
{
int i;
for (i=0;i<t*20;i++)
//cho 50ms
{
TMOD &= 0xf0;
TMOD |= 0x01;
ET0 = 0;
TH0 = 0x6f;
TL0 = 0xff;
TF0 = 0;
TR0 = 1;
while (!TF0);
TR0 = 0;
}
}


void nhanchuong()
{
ii dem=0;
long t;
P2=0x0f;
while (kochuong) P2=~P2;
while (dem<5)
{
dem++;
P2=dem;
cho(1);
      //tranh nhieu
while (!kochuong);
      //2s co chuong

t=0;
cho(2);
      //tranh nhieu
while ((kochuong)&&(t<50000))
t++;
if (t>=50000) //qua 4s ma
khong co chuong do lai
{dem=0;
P2=0x0f;
while (kochuong) P2=~P2;//doi
den khi co chuong lai
}
}
}


bit ok()
{
```

```
ii i;
npas=read(0);
if (npas!=sl) return 0;
for (i=1;i<=npas;i++)
{
if    (pass[i]!=read(i))
return 0;
}
return 1;
}

void delaydc()
{
ii i;
for (i=0;i<300;i++);
}

void mocua()
{
ii i;
for (i=0;i<240;i++)
{
P0&=0xf0;
P0+=1;
delaydc();
P0&=0xf0;
P0+=2;
delaydc();
P0&=0xf0;
P0+=4;
delaydc();
P0&=0xf0;
P0+=8;
delaydc();
}
}

void err()
{
EX0=0;
nhac2=1;
cho(5);
nhac2=0;
EX0=1;
}

void startPass()
{
EX0=0;
nhac1=1;
cho(5);
nhac1=0;
EX0=1;
}

void accept()
{
EX0=0;
nhac3=1;
cho(5);
nhac3=0;
EX0=1;
}

void changePass()
{
ii i;
switch (P2)
{
case 11: {sl=0;   startPass();
      break;}
case 12:  { EX0=0;

write(0,sl);
for (i=1;i<=sl;i++)

write(i,pass[i]);

accept();   xong=1;
      break;}
default: {  sl++;
      pass[sl]=P2;      }

}
}

void run()
{
switch (P2)
{
case 12: {EX0=0;
      accept();   mocua();
      xong=1;
      break;}     //mo cua
case 11: {buoc=1; sl=0;
      startPass();
      break;}     //thay mat
khau
}
}

void checkPass()
{     switch (P2)
{
case 11: {sl=0;   startPass();
      break;}
```

```
case 12:  { if (ok())
{buoc=2;
accept();}
else
{err();     xong=1;     }
break;}
default:
{sl++;
pass[sl]=P2;
delay(1000);
      //chong nhieu
}
}
}



void poll() interrupt NGAT0
      //cho IRQ\ xuong thap
{
P2=(ii)((ii)B3<<3)+((ii)B2<<2)
+((ii)B1<<1)+(ii)B0;
EX0=0;
P0=0;             //tat tieng
nhac
nwait=0;          //thoi gian
nguoi dung khong tac dong
switch (buoc)
{case 0: {checkPass();
      break;}
case 1:      {changePass();
      break;}
case 2:  {run();
      break;}
}
while (!kongat);
EX0=1;
}

void inPass()
{
ii i;
write(0,8);
for (i=1;i<=8;i++)
write(i,i);
}

void confirm()
{
ii i;
for (i=0;i<9;i++)
{
P2=read(i);
cho(1);
}
}

void timer1_init()
{
TMOD &= 0x0f;
TMOD |= 0x10;
TH1 = 0;
TL1 = 0;
TR1 = 1;
}

void wait() interrupt NGAT1
      //cho 65535 us
{
nwait++;
if (nwait>200)    xong=1;
}
main()
{
//NHAP MAT KHAU BAN DAU VAO BO
NHO
//inPass();
//confirm();

while(1)
{
xong=0;     buoc=0;
      nwait=0;
TR1=0;      EA=EX0=ET1=0;
P0=P1=P2=P3=0xff;
nhac1=nhac2=nhac3=0;
nhanchuong();
taigia=0;
EA=EX0=ET1=1;
timer1_init();
startPass();
while (!xong) P2=nwait;
}}
```