ãã¤ã¤ã¢ã³ãé»æ©æ ªå¼ä¼ç¤¾

**BÁO CÁO**

**Thiết kế sản phẩm thùng rác thông minh**

**Sử dụng cảm biến siêu âm HC SR-04**

*Thực hiện: Nguyễn Thị Ngọc Lan*

*Hướng dẫn: KS. Đào Minh Đức*

**MỤC LỤC**

Trang

[I. Phân tích đề tài 3](#_Toc14190023)

[II. Sơ đồ khối 3](#_Toc14190024)

[III. Các linh kiện cần dùng 4](#_Toc14190025)

[1. ATmega328PU 4](#_Toc14190026)

[2. Động cơ Servo 5](#_Toc14190027)

[3. Cảm biến siêu âm HC SR-04 6](#_Toc14190028)

[4. Một số linh kiện khác 7](#_Toc14190029)

[IV. Lập trình 8](#_Toc14190030)

[V. Thiết kế mạch nguyên lý 9](#_Toc14190031)

[VI. Hàn dán linh kiện 10](#_Toc14190032)

[VII. Hoàn thiện và đánh giá 10](#_Toc14190033)

## Phân tích đề tài

Mô hình thùng rác tự động đóng, mở nắp khi có người muốn bỏ rác vào thùng đồng thời cảnh báo cho người dùng biết khi thùng rác đầy sử dụng ATmega328PU nhận dữ liệu từ cảm biến siêu âm HC RS04 điều khiển động cơ servo.

## Sơ đồ khối

Chức năng các khối:

* Khối nguồn: cung cấp năng lượng cho toàn bộ mạch
* Khối xử lý: Nhận dữ liệu từ cảm biến và điều khiển các thiết bị thực thi
* Cảm biến: phát và thu sóng siêu âm từ đó tính toán giá trị khoảng cách từ nơi phát đến nơi bị phản xạ lại.
* Khối thực thi: thiết bị thực thi là servo có tác dụng cơ học đóng mở nắp thùng rác, đèn led báo hiệu lượng rác trong thùng.

Khối nguồn

Thực thi:

Servo, led

Khối xử lý

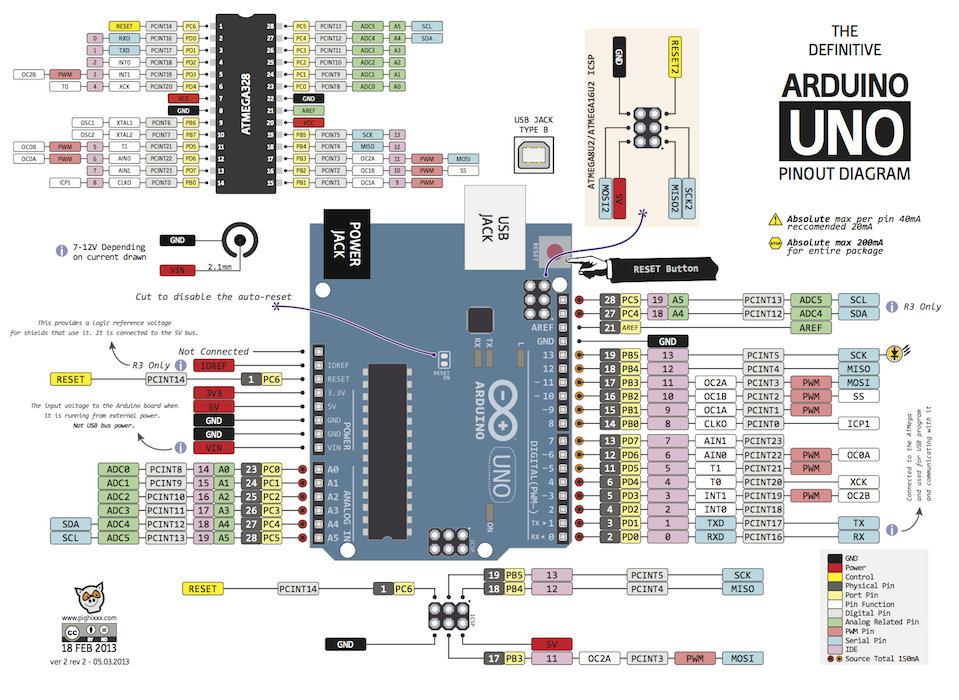
ATMega328

Cảm biến

HC SR04

## Các linh kiện cần dùng

### ATmega328PU

Là một chíp vi điều khiển được sản xuất bời hãng Atmel thuộc họ MegaAVR có sức mạnh hơn hẳn Atmega8. Atmega 328P là một bộ vi điều khiển 8 bít dựa trên kiến trúc RISC bộ nhớ chương trình 32KB ISP flash có thể ghi xóa hàng nghìn lần, 1KB EEPROM, một bộ nhớ RAM vô cùng lớn trong thế giới vi xử lý 8 bít (2KB SRAM) 

*Sơ đồ chân của Arduino và Vi xử lý ATmega 328*

Với 23 chân có thể sử dụng cho các kết nối vào hoặc ra I/O, 32 thanh ghi, 3 bộ timer/counter có thể lập trình, có các gắt nội và ngoại (2 lệnh trên một vector ngắt), giao thức truyền thông nối tiếp USART, SPI, I2C. Ngoài ra có thể sử dụng bộ biến đổi số tương tự 10 bít (ADC/DAC) mở rộng tới 8 kênh, khả năng lập trình được watchdog timer ( bộ đếm thời gian hoạt động liên tục nhằm tự động thực hiện một nhiệm vụ nào đó ), hoạt động với 5 chế độ nguồn, có thể sử dụng tới 6 kênh điều chế độ rộng xung (PWM), hỗ trợ bootloader.

*Bộ nhớ:*

+ *32KB bộ nhớ Flash*: những đoạn lệnh bạn lập trình sẽ được lưu trữ trong bộ nhớ Flash của vi điều khiển. Thường thì sẽ có khoảng vài KB trong số này sẽ được dùng cho bootloader.

+ *2KB cho SRAM* (Static Random Access Memory): giá trị các biến bạn khai báo khi lập trình sẽ lưu ở đây. Khai báo càng nhiều biến thì càng cần nhiều bộ nhớ RAM. Khi ngắt nguồn thì dữ liệu cũng sẽ mất.

+ *1KB cho EEPROM* (Electrically Eraseble Programmable Read Only Memory): đây giống như một chiếc ổ cứng mini – nơi có thể đọc và ghi dữ liệu vào đây.

### Động cơ Servo



*Hình ảnh động cơ Servo*

Servo là một dạng động cơ điện đặc biệt. Không giống như động cơ thông thường cứ cắm điện vào là quay liên tục, servo chỉ quay khi được điều khiển (bằng xung PPM) với góc quay nằm trong khoảng bất kì từ 0o - 180o. Mỗi loại servo có kích thước, khối lượng và cấu tạo khác nhau.

Động cơ servo được thiết kế những hệ thống hồi tiếp vòng kín. Tín hiệu ra của động cơ được nối với một mạch điều khiển. Khi động cơ quay, vận tốc và vị trí sẽ được hồi tiếp về mạch điều khiển này. Nếu có bầt kỳ lý do nào ngăn cản chuyển động quay của động cơ, cơ cấu hồi tiếp sẽ nhận thấy tín hiệu ra chưa đạt được vị trí mong muốn. Mạch điều khiển tiếp tục chỉnh sai lệch cho động cơ đạt được điểm chính xác.

Động cơ servo cũng được chia làm nhiều loại, phụ thuộc vào góc quay tối đa của chúng. 2 loại phổ biến ta hay sử dụng là:

Động cơ servo quay 180°: Futaba S3003, MG90[S] ...

Động cơ servo quay 360°: MG995, MG996R ...

Chân Đỏ: nhận điện nguồn, tuỳ vào loại động cơ mà giá trị này có thể khác nhau

Chân Đen: nối với cực âm của mạch

Chân Vàng: nhận tín hiệu từ mạch điều khiển

### Cảm biến siêu âm HC SR-04

Cảm biến siêu âm SR04 hoạt động bằng cách phát 1 tín hiệu siêu âm (tần số 40Khz) và bắt đầu đếm thời gian nhận được tín hiệu phản hồi. Khoảng cách do cảm biến siêu âm SR-04 được tính từ tốc độ và thời gian phản hồi.



*Cảm biến siêu âm HC SR-04*

Xét về mặt nguyên lí hoạt động thì cảm biến sẽ phát ra 1 làn sóng siêu âm và thu ngược về. Sóng siêu âm khi gặp vật cản sẽ tự phản ngược về trong lúc thu về thì ta có được thời gian từ khi phát đến lúc thu. Qua đó trên tính toán vật lí ta sẽ có được khoảng cách từ cảm biến đến vật cản.

Tốc độ âm thanh trong không khí là 340m/s (hằng số) tương đương 29.412microSecond/cm. Sau khi thu được thời gian đi và về thì chia cho /2/29.412 sẽ ra được khoảng cách (cm). 340m/s ở đây là trong không khí nhiệt độ bình thường, ở môi trường khác các cần tính toán lại.

HC-SR04 gồm 4 chân: VCC (5V), GND, Trig, Echo. Để sử dụng cảm biến, các bạn phát ra 1 xung ngắn từ chân Trig. sau đó bật HIGH chân Echo đến khi nhận lại tín hiệu đã phát. Chiều rộng của xung sẽ bằng với thời gian sóng siêu âm thu được và quay lại.

### Một số linh kiện khác

* Điện trở
* Tụ gốm 22pF
* Tụ phân cực 10uF 16V
* Công tắc
* Nút nhấn
* Led
* Thạch anh 16MHz
* Board mạch đục lỗ
* Header 1 hàng
* Jump cắm
* Pin
* Dây dẫn
* …

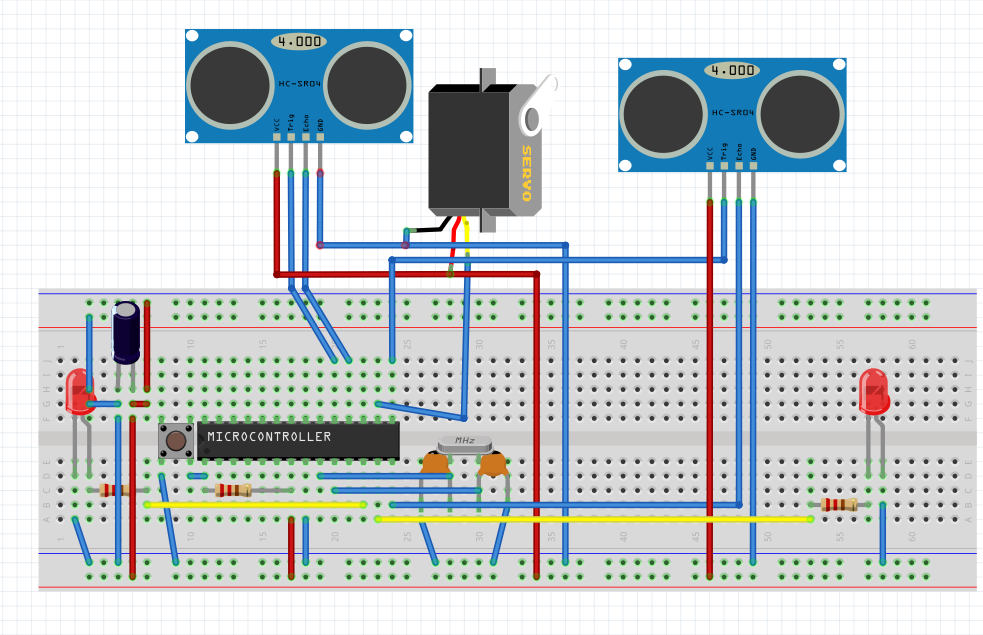
## Lập trình

1. #include<Servo.h>
2. int trig = 13;
3. int echo = 12;
4. int trig2 = 9;
5. **int echo2 = 8;**
6. int led1 = 7;
7. int led2 = 6;
8. int servoPin = 10;
9. Servo servo;
11. void setup(){
12. Serial.begin(9600);
13. servo.attach(servoPin);
14. pinMode(trig, OUTPUT);
15. **pinMode(echo, INPUT);**
16. pinMode(trig2, OUTPUT);
17. pinMode(echo2, INPUT);
18. pinMode(led1, OUTPUT);
19. pinMode(led2, OUTPUT);
20. **}**
22. void loop(){
23. distance();
24. *// Serial.println(distance());*
25. ***// Serial.println("cm");***
26. *// delay(200);*
27. if(distance()<30){
28. do{
29. servo.write(90);
30. **distance();**
31. }while(distance()<30);
32. delay(5000);
33. for(int i=90;i>0;i--){
34. servo.write(i);
35. **delay(20);**
36. }
37. }
38. else {
39. servo.write(0);
40. **}**
41. *// Serial.print(distance2());*
42. distance2();
43. if(distance2()<15){
44. digitalWrite(led1,HIGH);
45. **}**
46. else{
47. digitalWrite(led1,LOW);
48. }
49. if(distance2()<8){
51. digitalWrite(led2,HIGH);
52. }
53. else{
54. digitalWrite(led2,LOW);
55. **}**
56. }
57. float distance(){
58. digitalWrite(trig, HIGH);
59. delayMicroseconds(5);
60. **digitalWrite(trig,LOW);**
61. int timer = pulseIn(echo, HIGH);
62. return timer/58.3f;
63. }
64. float distance2(){
65. **digitalWrite(trig2, HIGH);**
66. delayMicroseconds(5);
67. digitalWrite(trig2,LOW);
68. int timer = pulseIn(echo2, HIGH);
69. return timer/58.3f;
70. **}**

Thuật toán: Khai báo thư viện, các chân và giá trị khởi tạo. Trong vòng lặp, vi điều khiển phát tín hiệu ở chân trig và thu tín hiệu song siêu âm ở chân echo. Sau đó với thông số kỹ thuật của cảm biến sẽ tính toán khoảng cách từ nơi phát ra tín hiệu là cảm biến đến nơi gặp vật cản khiến sóng phản xạ lại, tính ra đơn vị đo khoảng cách. Khi nhận biết có tác động ở gần hệ thống sẽ quay servo điều kiển mở nắp thùng rác. Sau đó đợi 5s để bỏ rác vào thùng, thùng rác sẽ tự động nhẹ nhàng đóng nắp lại khi cho servo về vị trí ban đầu. Ở trạng thái không có tác động đến cảm biến, servo ở trạng thái ban đầu.

Khi lượng rác trong thùng tăng lên, cảm biến siêu âm sẽ thu thập dữ liệu khoảng cách từ rác đến nắp nhờ bộ xử lý sẽ báo hiệu lại cho người dùng thể hiện trên led. Khi lượng rác cách nắp 15cm, hệ thống sẽ cảnh báo mức 1, led xanh sáng. Khi lượng rác cách nắp 8cm, hệ thống sẽ cảnh báo mức 2, cả led xanh và led đỏ sáng.

## Thiết kế mạch nguyên lý



*Sơ đồ mạch mô phỏng trên fritzing*

## Sản phẩm



## Hoàn thiện và đánh giá

Thiết bị chưa có được nhiều chức năng, cần làm cho thùng rác tự động và thông minh hơn, chưa thiết thực lắm.