# Chương 1: Mở đầu

# Chương 2: Giới thiệu các thành phần phần cứng:

## Arduino

* 1. Giới thiệu Arduino

**Lịch sử ra đời**

* Tiền thân của Arduino là Basic Stamps*.* Basic Stamps là 1 bộ vi điều khiển ( microcontroller ) sử dụng bộ thông dịch chuyên dụng BASIC ( PBASIC ) xây dựng nên ROM. Giá của Basic Stamps rất cao , lên đến 100$.
* Arduino được khởi động vào năm 2005 như là một dự án dành cho sinh viên Interaction Design Institute Ivrea (Viện thiết kế tương tác Ivrea) tại Ivrea, Italy. Dự án được đứng đầu bởi Massimo Banzi, một giảng viên tại Ivrea.
* Lý thuyết phần cứng được đóng góp bởi một sinh viên Hernando Barragan người Colombia. Sau khi nền tảng Wiring hoàn thành, các nhà nghiên cứu đã không làm việc tiếp trên nền tảng này. Thay vì vậy, họ sao chép mã nguồn của Wiring để bắt đầu 1 dự án khác với cái tên là Arduino. Dự án này sau đó đã giúp nó nhẹ hơn, rẻ hơn, và khả dụng đối với cộng đồng mã nguồn mở.
* Cái tên "Arduino" đến từ một quán bar tại Ivrea, nơi một vài nhà sáng lập của dự án này thường xuyên gặp mặt. Bản thân quán bar này được lấy tên là Arduino, theo tên của Bá tước của Ivrea, và là vua của Italy từ năm 1002 đến 1014.
* Viện Ivrea cuối cùng bị đóng cửa, vì vậy các nhà nghiên cứu, một trong số đó là David Cuarlielles, đã phổ biến ý tưởng này.

**Cấu trúc**

* Một mạch Arduino bao gồm **một vi điều khiển AVR** với nhiều linh kiện bổ sung giúp dễ dàng lập trình và có thể mở rộng với các mạch khác.
* Arduino cho phép người dùng kết nối với CPU của board với các module thêm vào có thể dễ dàng chuyển đổi, được gọi là *shield*. Vài shield truyền thông với board Arduino trực tiếp thông qua các chân khác nhau, nhưng nhiều shield được định địa chỉ thông qua **serial bus I²C** (nhiều shield có thể được xếp chồng và sử dụng dưới dạng song song).
* Arduino chính thức thường sử dụng các dòng **chip megaAVR** (ATmega8, ATmega168, ATmega328, ATmega1280, và ATmega2560).
* Hầu hết các mạch gồm **một bộ điều chỉnh tuyến tính 5V** và **một thạch anh dao động 16 MHz** (hoặc **bộ cộng hưởng ceramic** trong một vài biến thể), mặc dù một vài thiết kế như LilyPad chạy tại 8 MHz và bỏ qua bộ điều chỉnh điện áp onboard do hạn chế về kích cỡ thiết bị.
* Theo nguyên tắc, khi sử dụng ngăn xếp phần mềm Arduino, tất cả các board được lập trình thông qua một kết nối RS-232, nhưng cách thức thực hiện lại tùy thuộc vào đời phần cứng. Các board Serial Arduino có chứa **một mạch chuyển đổi giữa RS232 sang TTL**. Các board Arduino hiện tại được lập trình thông qua cổng USB, thực hiện thông qua chip chuyển đổi USB-to-serial như là FTDI FT232.
* Board Arduino sẽ đưa ra hầu hết **các chân I/O của vi điều khiển** để sử dụng cho những mạch ngoài. Những chân này được thiết kế nằm phía trên mặt board, thông qua các header cái 0.10-inch (2.5 mm). Các board Arduino Nano, và Arduino-compatible Bare Bones Board và Boarduino có thể cung cấp các chân header đực ở mặt trên của board dùng để cắm vào các breadboard.
* Có nhiều biến thể như Arduino-compatible và Arduino-derived. Một vài trong số đó có chức năng tương đương với Arduino và có thể sử dụng để thay thế qua lại. Nhiều mở rộng cho Arduino được thực thiện bằng cách thêm vào các driver đầu ra, thường sử dụng trong các trường học để đơn giản hóa các cấu trúc của các 'con rệp' và các robot nhỏ. Những board khác thường tương đương về điện nhưng có thay đổi về hình dạng-đôi khi còn duy trì độ tương thích với các shield, đôi khi không. Vài biến thể sử dụng bộ vi xử lý hoàn toàn khác biệt, với các mức độ tương thích khác nhau.
  1. Arduino Mega
* 54 chân digital (15 có thể được sử dụng như các chân PWM)
* 16 đầu vào analog
* 4 UARTs (cổng nối tiếp phần cứng)
* 1 thạch anh 16 MHz
* 1 cổng kết nối USB
* 1 jack cắm điện
* 1 đầu ICSP
* 1 nút reset

Nó chứa **tất cả** mọi thứ cần thiết để hỗ trợ các vi điều khiển. Arduino Mega2560 khác với tất cả các vi xử lý trước giờ vì không sử dụng FTDI chip điều khiển chuyển tín hiệu từ USB để xử lý. Thay vào đó, nó sử dụng ATmega16U2 lập trình như là một công cụ chuyển đổi tín hiệu từ USB. Ngoài ra, Arduino Mega2560 cơ bản vẫn giống Arduino Uno R3, chỉ khác số lượng chân và nhiều tính năng mạnh mẽ hơn, nên các bạn vẫn có thể lập trình cho con vi điều khiển này bằng chương trình lập trình cho Arduino Uno R3.



## Module điều khiển động cơ L298 v3

* 1. Giới thiệu
* Module sử dụng IC L298, có chức năng điều khiển động cơ DC, động cơ bước
* Có thể đảo chiều động cơ
* Sử dụng thông qua vi điều khiển hoặc L297
* DC: +5-35V; I <= 2A
* kích thước: 55mmx49mmx33mm
  1. Cách đi dây với Arduino



* 1. Các lệnh điều khiển cơ bản

void DieuKhien(){

//2 trường hợp: khi gặp chướng ngại vật và trường hợp còn lại, mỗi trường hợp áp dụng các luật mờ tương ứng, kết quả sau khi giải

//mờ là vận tốc đích Vo, góc rẽ trái, phải (RL,RR)

//điều khiển thử nghiệm:

if(FF>0.6){

//di thang

RR=0;RL=0; Vo=0.2;

}

else if(FL>1){

RR=0;RL=1; Vo=0.1;

}

else if(FR>1){

RR=1;RL=0;Vo=0.1;

}

if(FF<0.15)

Vo=-1;

if(FF>0.6&&FR>1&&FL>1){

}

Serial.print("DeltaDirection= ");

Serial.println(heading-orginalDirection);

Serial.print("RR= ");

Serial.println(RR);

Serial.print("RL= ");

Serial.println(RL);

Serial.print("Vo= ");

Serial.println(Vo);

}

void VanHanh(){

//gồm 2 chế độ : chế độ rẽ và chế độ đi thẳng

//chế độ rẽ (nếu góc RL,RR đủ lớn) : căn cứ trên vận tốc đích Vo và các góc rẽ mà thiết lập điện áp trên 2 bánh

//chế độ đi thẳng: căn cứ trên gia tốc góc để giữ cho xe chạy thẳng với tốc độ tối đa, sử dụng PCI đơn giản

HR = RL - RR;

//Vo = 130;

int Vmin=80;

if (abs(HR) < 0.05) {

Va = Vb = V = Vo \* (255 - Vmin) + Vmin;

}

if (abs(HR) > 0.05) {

if (HR < 0) {

Va = Vo \* (225 - Vmin) + Vmin;

Vb = Va \* (1 - abs(HR));

}

else if (HR > 0) {

Vb = Vo \* (225 - Vmin) + Vmin;

Va = Vb \* (1 - abs(HR));

}

}

if (abs(HR) < 0.05) {

digitalWrite(INA, HIGH);

digitalWrite(INC, HIGH);

digitalWrite(INB, LOW);

digitalWrite(IND, LOW);

analogWrite(SPA, Va);

analogWrite(SPB, Vb);

}

if (abs(HR) > 0.05) {

if (HR < 0) {

digitalWrite (INC, HIGH);

digitalWrite (INA, HIGH);

digitalWrite (INB, LOW);

digitalWrite (IND, LOW);

analogWrite(SPA, Va);

analogWrite(SPB, Vb);

}

else if (HR > 0) {

digitalWrite (INC, HIGH);

digitalWrite (INA, HIGH);

digitalWrite (INB, LOW);

digitalWrite (IND, LOW);

analogWrite(SPA, Va);

analogWrite(SPB, Vb);

}

}

if(Vo<0){

digitalWrite (INC, LOW);

digitalWrite (INA, LOW);

digitalWrite (INB, HIGH);

digitalWrite (IND, HIGH);

analogWrite(SPA, Vmin);

analogWrite(SPB, Vmin);

}

Serial.print("HR= ");

Serial.println(HR);

Serial.print("Va= ");

Serial.println(Va);

Serial.print("Vb= ");

Serial.println(Vb);

}

## La bàn số LSM303

* 1. Giới thiệu

Cảm biến la bàn số và gia tốc GY-511 LSM303DLHC có khả năng đo 3 trục gia tốc hướng (Accelerometer) và 3 trục từ trường (Magnetometer), thường được dùng để xác định hướng chuyển động và hướng từ trường của trái đất nhằm xác định phương hướng với độ chính xác cao.

* Điện áp sử dụng: 3~5VDC
* Điện áp giao tiếp: 3~5VDC
* Chuẩn giao tiếp: I2C
* 3 trục từ trường và 3 trục gia tốc
* Full scale range of ± 1.3 to ±8.1 gauss magnetic field
* 16 bit dữ liệu đầu ra
* Chế độ bật /tắt nguồn
* 6DOF orientation detection
* Kích thước: 14.5 x 20.5mm
  1. Cách đi dây với Arduino



* 1. Lấy dữ liệu từ LSM303

sử dụng thư viện

#include <Adafruit\_Sensor.h>

#include <Adafruit\_LSM303\_U.h>

float thuHuongDi(){

//thu nhận dữ liệu từ la bàn số, tính toán hướng hiện tại D và góc lệch so với hướng đích DD

/\* Get a new sensor event \*/

Serial.print("Compass Heading: ");

sensors\_event\_t event;

mag.getEvent(&event);

float Pi = 3.14159f;

// Calculate the angle of the vector y,x

heading = (atan2(event.magnetic.y,event.magnetic.x) \* 180) / Pi;

// Normalize to 0-360

if (heading < 0)

{

heading = 360 + heading;

}

Serial.print("Compass Heading: ");

Serial.println(heading);

return heading;

//delay(500);

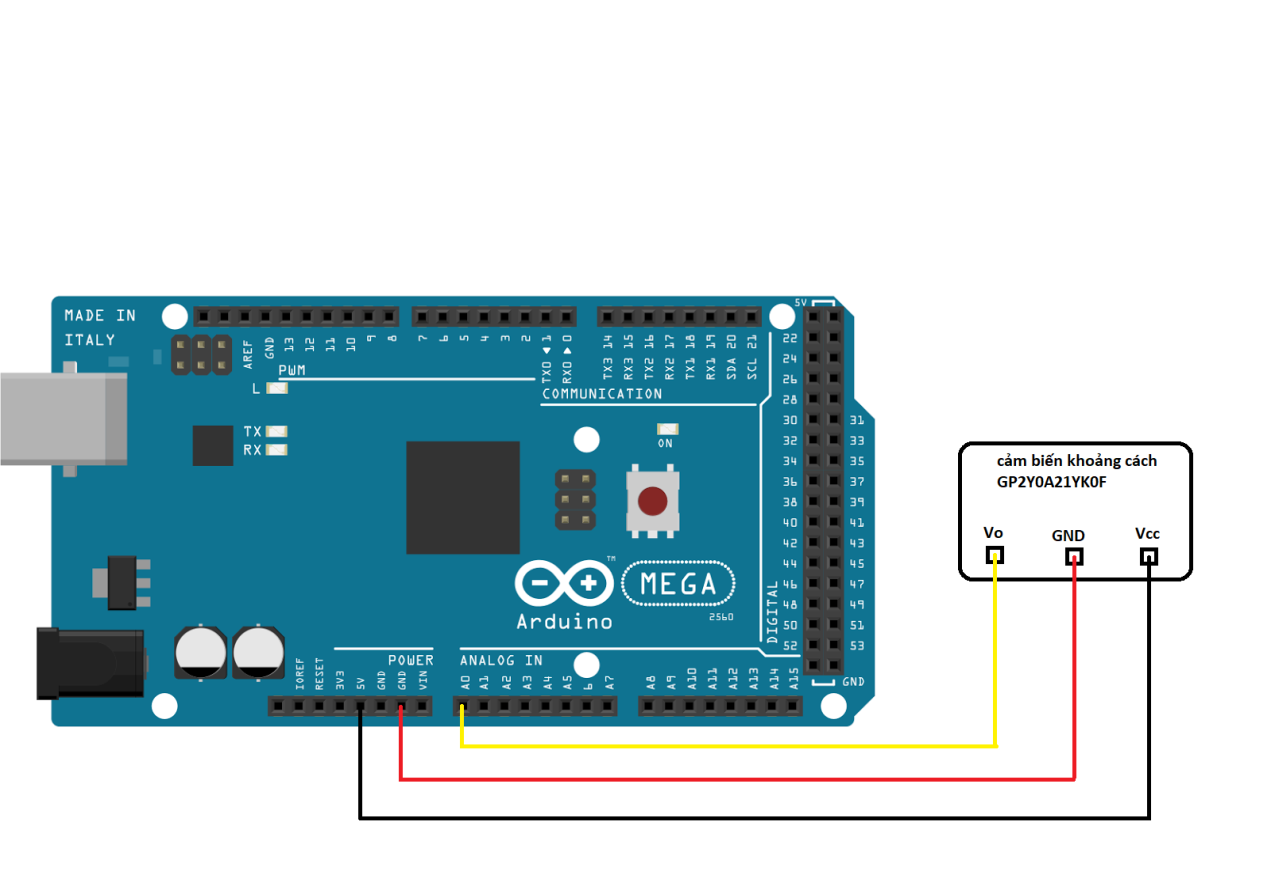
}

## Cảm biến khoảng cách GP2Y0A21YK0F

* 1. Giới thiệu

GP2Y0A21YK0F là cảm biến dùng tia hồng ngoại để đo khoảng cách (tối đa 80cm). Giá trị trả về là analog nên rất dễ sử dụng.

* Điện áp hoạt động: 4.5 - 5.5 V
* Dòng điện tiêu thụ: 30mA
* Khoảng cách đo: 10 – 80 cm
* Ngõ ra: Analog
* Tốc độ hồi lấy mẫu: 38 ± 10 ms
* Kích thước: 44.5 x 18.9 x 13.5
  1. Cách đi dây



* 1. Lấy dữ liệu

void thuKhoangCach(){

//thu nhận dữ liệu từ sensor khoảng cách, cho ra các khoảng cách FF,FR,FL tính bằng m

//front

int i;

i=analogRead(redpin);

FF=((float)6762/(i-9))-4/100;

FF=(float)FF/100;

/\* In kết quả ra Serial Monitor \*/

Serial.print("front= ");

Serial.print(FF);

Serial.println("m");

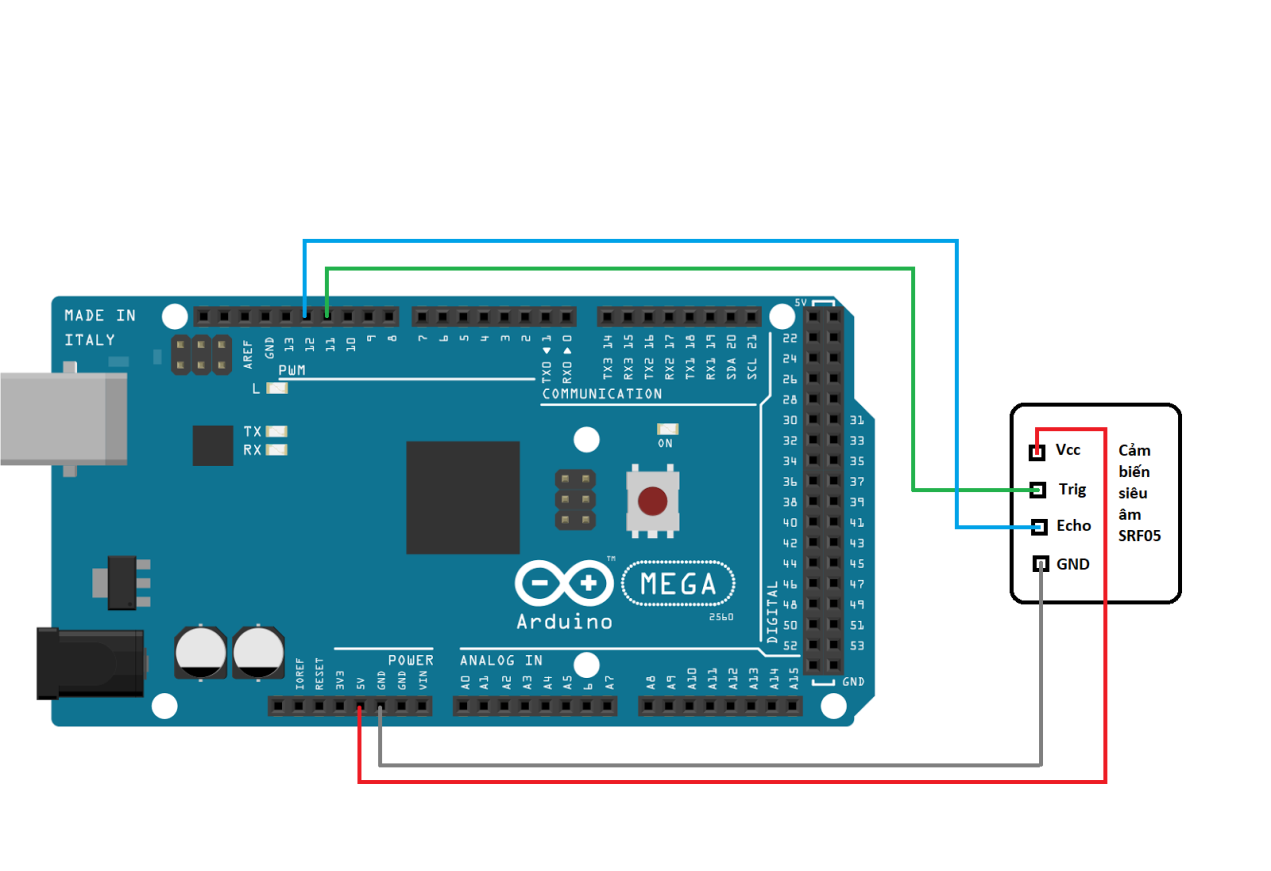
}

## Cảm biến siêu âm SRF05

* 1. Giới thiệu

Cảm biến SRF05 là phiên bản nâng cấp của SRF04, cảm biến dùng để nhận biết khoảng cách đến vật cản nhờ sóng siêu âm phát ra từ cảm biến và nhận về (Quãng đường = vận tốc x thời gian / 2).

* Điện áp hoạt động: 5VDC
* Dòng cấp: < 2mA
* Góc quét:<15 độ
* Tín hiệu: TTL
* Phát hiện vật cản trong khoảng: 2-300cm
* Kích thước: 43mm x 20mm x 17mm
  1. Cách đi dây với Arduino



* 1. Lấy dữ liệu

void thuKhoangCach(){

//thu nhận dữ liệu từ sensor khoảng cách, cho ra các khoảng cách FF,FR,FL tính bằng m

unsigned long duration\_left, duration\_right; // biến đo thời gian

/\* Phát xung từ chân trig \*/

digitalWrite(trig\_left,0); // tắt chân trig

delayMicroseconds(2);

digitalWrite(trig\_left,1); // phát xung từ chân trig

delayMicroseconds(5); // xung có độ dài 5 microSeconds

digitalWrite(trig\_left,0); // tắt chân trig

/\* Tính toán thời gian \*/

// Đo độ rộng xung HIGH ở chân echo.

duration\_left = pulseIn(echo\_left,HIGH);

// Tính khoảng cách đến vật.

FL = (duration\_left/2/29.412/100);

//right

digitalWrite(trig\_right,0);

delayMicroseconds(2);

digitalWrite(trig\_right,1);

delayMicroseconds(5);

digitalWrite(trig\_right,0);

duration\_right = pulseIn(echo\_right,HIGH);

FR = (duration\_right/2/29.412/100);

Serial.print("left= ");

Serial.print(FL);

Serial.println("m");

Serial.print("right= ");

Serial.print(FR);

Serial.println("m");

}

## GPS GPS NEO-7M

* 1. Giới thiệu
  2. Cách đi dây với Arduino
  3. Lấy dữ liệu

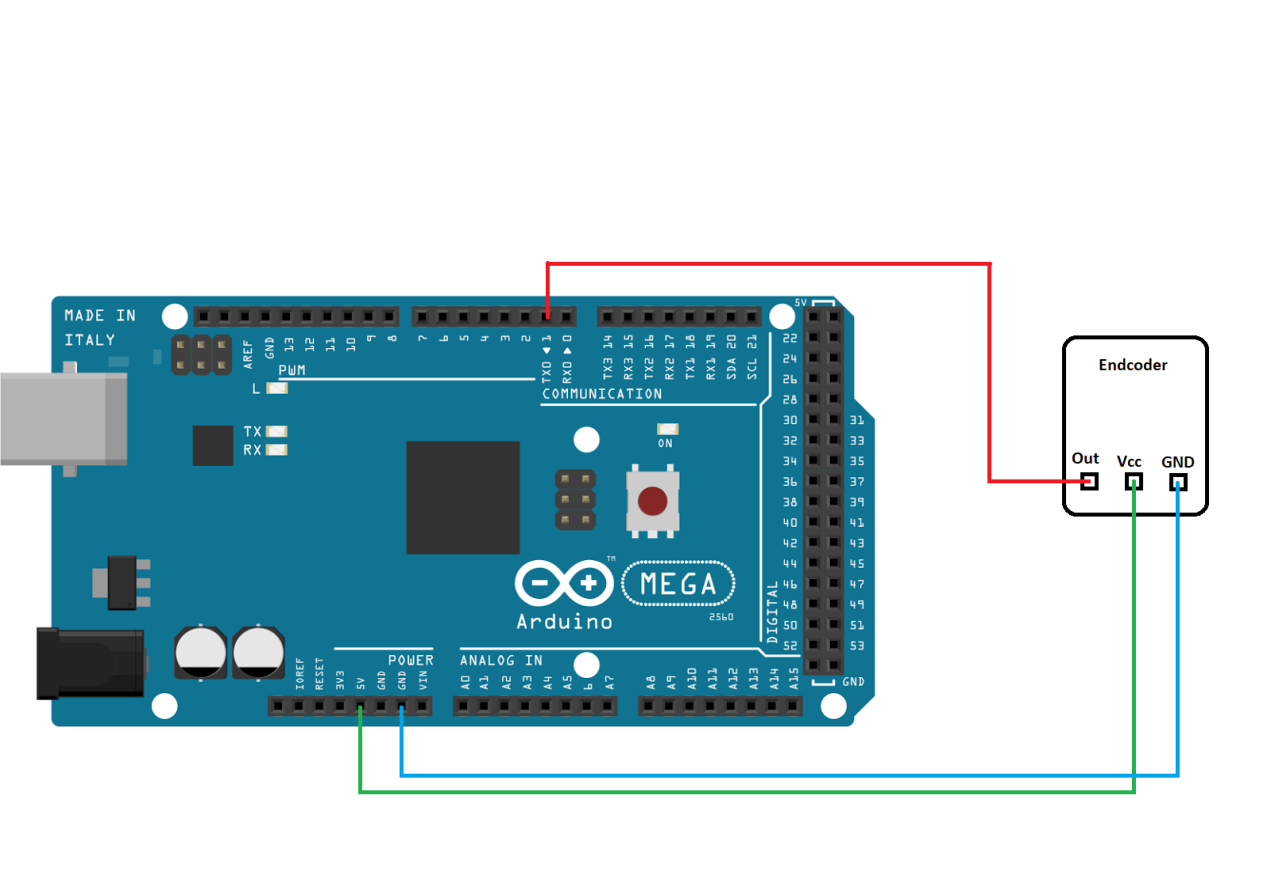
## Encoder đo vận tốc

* 1. Giới thiệu

Endcoder là một phần tử được ứng dụng rộng rãi trong các hệ thống điều khiển tự động, hay đơn giản như gắn trong bánh xe hơi để báo vận tốc.

Nguyên lý cơ bản của encoder, đó là một đĩa tròn xoay, quay quanh trục. Trên đĩa có các lỗ (rãnh). Người ta dùng một đèn led để chiếu lên mặt đĩa. Khi đĩa quay, chỗ không có lỗ (rãnh), đèn led không chiếu xuyên qua được, chỗ có lỗ (rãnh), đèn led sẽ chiếu xuyên qua. Khi đó, phía mặt bên kia của đĩa, người ta đặt một con mắt thu. Với các tín hiệu có, hoặc không có ánh sáng chiếu qua, người ta ghi nhận được đèn led có chiếu qua lỗ hay không.Số xung đếm được và tăng lên nó tính bằng số lần ánh sáng bị cắt!  
Như vậy là encoder sẽ tạo ra các tín hiệu xung vuông và các tín hiệu xung vuông này được cắt từ ánh sáng xuyên qua lỗ. Nên tần số của xung đầu ra sẽ phụ thuộc vào tốc độ quay của tấm tròn đó. Đối với encoder mình đang dùng thì nó có 2 tín hiệu ra lệch pha nhau 90. Hai tín hiệu này có thể xác định được chiều quay của động cơ.

* 1. Cách đi dây với Arduino



* 1. Lấy dữ liệu

void thuVanToc(){

//thu nhận tốc độ hiện tại V của xe bằng encoder, quy ra m/s

float n=(float)count/20;

//float v=1000/(millis()-lasttime)\*n;

float v=20/(100\*(millis()-lasttime))\*n;

lasttime=millis();

Serial.println(count);

count=0;

Serial.println(v);

}

# Chương 3: Ứng dụng điều khiển mờ trong xe tự lái

1. Giới thiệu về điều khiển mờ

Logic mờ (tiếng Anh: *Fuzzy logic*) được phát triển từ lý thuyết tập mờ để thực hiện lập luận một cách xấp xỉ thay vì lập luận chính xác theo lôgic vị từ cổ điển. Logic mờ có thể được coi là mặt ứng dụng của lý thuyết tập mờ để xử lý các giá trị trong thế giới thực cho các bài toán phức tạp

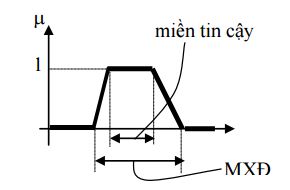
Lôgic mờ cho phép độ liên thuộc có giá trị trong khoảng đóng 0 và 1, và ở hình thức ngôn từ, các khái niệm không chính xác như "hơi hơi", "gần như", "khá là" và "rất". Cụ thể, nó cho phép quan hệ thành viên không đầy đủ giữa thành viên và tập hợp.

Logic mờ được công bố lần đầu tiên tại Mỹ vào năm 1965 do giáo sư Lotfi Zadeh. Kể từ đó, logic mờ đã có nhiều phát triển qua các chặng đường sau : phát minh ở Mỹ, áp dụng ở Châu Âu và đưa vào các sản phẩm thương mại ở Nhật. Ứng dụng đầu tiên của logic mờ vào công nghiệp được thực hiện ở Châu Âu, khoảng sau năm 1970. Tại trường Queen Mary ở Luân Đôn – Anh, Ebrahim Mamdani dùng logic mờ để điều khiển một máy hơi nước mà trước đây ông ấy không thể điều khiển được bằng các kỹ thuật cổ điển. Và tại Đức, Hans Zimmermann dùng logic mờ cho các hệ ra quyết định. Liên tiếp sau đó, logic mờ được áp dụng vào các lĩnh vực khác như điều khiển lò xi măng, … nhưng vẫn không được chấp nhận rộng rãi trong công nghiệp. Kể từ năm 1980, logic mờ đạt được nhiều thành công trong các ứng dụng ra quyết định và phân tích dữ liệu ở Châu Âu. Nhiều kỹ thuật logic mờ cao cấp được nghiên cứu và phát triển trong lĩnh vực này. Cảm hứng từ những ứng dụng của Châu Âu, các công ty của Nhật bắt đầu dùng logic mờ vào kỹ thuật điều khiển từ năm 1980. Nhưng do các phần cứng chuẩn tính toán theo giải thuật logic mờ rất kém nên hầu hết các ứng dụng đều dùng các phần cứng chuyên về logic mờ. Một trong những ứng dụng dùng logic mờ đầu tiên tại đây là nhà máy xử lý nước của Fuji Electric vào năm 1983, hệ thống xe điện ngầm của Hitachi vào năm 1987. Những thành công đầu tiên đã tạo ra nhiều quan tâm ở Nhật. Có nhiều lý do để giải thích tại sao logic mờ được ưa chuộng. Thứ nhất, các kỹ sư Nhật thường bắt đầu từ những giải pháp đơn giản, sau đó mới đi sâu vào vấn đề. Phù hợp với việc logic mờ cho phép tạo nhanh các bản mẫu rồi tiến đến việc tối ưu. Thứ hai, các hệ dùng logic mờ đơn giản và dễ hiểu. Sự “thông minh” của hệ không nằm trong các hệ phương trình vi phân hay mã nguồn. Cũng như việc các kỹ sư Nhật thường làm việc theo tổ, đòi hỏi phải có một giải pháp để mọi người trong tổ đều hiểu được hành vi của hệ thống, cùng chia sẽ ý tưởng để tạo ra hệ. Logic mờ cung cấp cho họ một phương tiện rất minh bạch để thiết kế hệ thống. Và cũng do nền văn hóa, người Nhật không quan tâm đến logic Boolean hay logic mờ; cũng như trong tiếng Nhật , từ “mờ‟ không mang nghĩa tiêu cực. Do đó, logic mờ được dùng nhiều trong các ứng dụng thuộc lĩnh vực điều khiển thông minh hay xử lý dữ liệu. Máy quay phim và máy chụp hình dùng logic mờ để chứa đựng sự chuyên môn của người nghệ sĩ nhiếp ảnh. Mitsubishi thông báo về chiếc xe đầu tiên trên thế giới dùng logic mờ trong điều khiển, cũng như nhiều hãng chế tạo xe khác của Nhật dùng logic mờ trong một số thành phần. Trong lĩnh vực tự động hóa, Omron Corp. có khoảng 350 bằng phát minh về logic mờ. Ngoài ra, logic mờ cũng được dùng để tối ưu nhiều quá trình hóa học và sinh học.

Điều khiển mờ:

+ Định nghĩa tập mờ

Tập mờ F xác định trên tập kinh điển B là một tập mà mỗi phần tử của nó là một cặp giá trị (x,µF(x)), với x∈ X và µF(x) là một ánh xạ : µF(x) : B → [0 1] Điều khiển mờ trong đó : µF gọi là hàm thuộc , B gọi là tập nền.



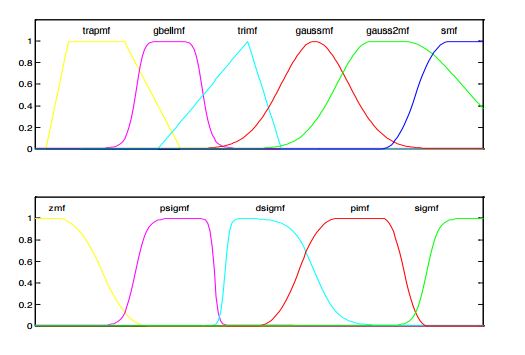
+ Các thuật ngữ trong logic mờ

Độ cao tập mờ F là giá trị h = SupµF(x), trong đó supµF(x) chỉ giá trị nhỏ nhất trong tất cả các chặn trên của hàm µF(x).

Miền xác định của tập mờ F, ký hiệu là S là tập con thoả mãn : S = SuppµF(x) = { x∈B | µF(x) > 0}

Miền tin cậy của tập mờ F, ký hiệu là T là tập con thoả mãn : T = { x∈B | µF(x) = 1 }

Các dạng hàm thuộc (membership function) trong logic mờ. Có rất nhiều dạng hàm thuộc như : Gaussian, PI-shape, S-shape, Sigmoidal, Z-shape …



+ Biến ngôn ngữ

Biến ngôn ngữ là phần tử chủ đạo trong các hệ thống dùng logic mờ. Ở đây các thành phần ngôn ngữ của cùng một ngữ cảnh được kết hợp lại với nhau. Để minh hoạ về hàm thuộc và biến ngôn ngữ ta xét ví dụ sau : Xét độ ẩm đo được , ta có thể phát biểu: - Rất cao - Cao - Trung bình - Thấp - Rất thấp . Những phát biểu như vậy gọi là biến ngôn ngữ của tập mờ.

**Các phép toán trên tập mờ**

Cho X, Y là hai tập mờ trên không gian nền B, có các hàm thuộc tương ứng là µX, µY. Khi đó: - Phép hợp hai tập mờ: X∪Y

+ Theo luật Max µX∪Y(b) = Max{ µX(b) , µY(b) }

+ Theo luật Sum µX∪Y(b) = Min{ 1, µX(b) + µY(b) }

+ Tổng trực tiếp µX∪Y(b) = µX(b) + µY(b) - µX(b).µY (b)

- Phép giao hai tập mờ: X∩Y

+ Theo luật Min µX∪Y(b) = Min{ µX(b) , µY(b) }

+ Theo luật Lukasiewicz µX∪Y(b) = Max{0, µX(b)+µY(b)-1}

+ Theo luật Prod µX∪Y(b) = µX(b).µY(b)

- Phép bù tập mờ: X µ (b) = 1- µX(b)

**Luật hợp thành**

Mệnh đề hợp thành

Ví dụ điều khiển mực nước trong bồn chứa, ta quan tâm đến 2 yếu tố:

+ Mực nước trong bồn L = {rất thấp, thấp, vừa}

+ Góc mở van ống dẫn G = {đóng, nhỏ, lớn}

Ta có thể suy diễn cách thức điều khiển như thế này:

Nếu mực nước = rất thấp Thì góc mở van = lớn

Nếu mực nước = thấp Thì góc mở van = nhỏ

Nếu mực nước = vừa Thì góc mở van = đóng

Trong ví dụ trên ta thấy có cấu trúc chung là “Nếu A thì B”. Cấu trúc này gọi là mệnh đề hợp thành, A là mệnh đề điều kiện, C = A⇒B là mệnh đề kết luận.

Định lý Mamdani: “Độ phụ thuộc của kết luận không được lớn hơn độ phụ thuộc điều kiện” Nếu hệ thống có nhiều đầu vào và nhiều đầu ra thì mệnh đề suy diễn có dạng tổng quát như sau: If N = ni and M = mi and … Then R = ri and K = ki and ….

**Luật hợp thành mờ**

Luật hợp thành là tên gọi chung của mô hình biểu diễn một hay nhiều hàm thuộc cho một hay nhiều mệnh đề hợp thành.

Các luật hợp thành cơ bản Điều khiển mờ

+ Luật Max – Min

+ Luật Max – Prod

+ Luật Sum – Min

+ Luật Sum – Prod

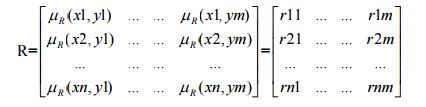
Thuật toán xây dựng mệnh đề hợp thành cho hệ SISO

Luật mờ cho hệ SISO có dạng “If A Then B”

Chia hàm thuộc µA(x) thành n điểm xi, i=1,2….,n

Chia hàm thuộc µB(y) thành m điểm yj , j = 1,2,…,m

Xây dựng ma trận quan hệ R



Hàm thuộc µB’(y) đầu ra ứng với giá trị rõ đầu vào xk có giá trị µB’(y) = a T .R , với a T = { 0,0,0,…,0,1,0….,0,0 }. Số 1 ứng với vị trí thứ k.

Trong trường hợp đầu vào là giá trị mờ A’ thì µB’(y) là: µB’(y) = { l1,l2,l3,…,lm } với lk=maxmin{ai,rik }.

+ Giải mờ

Giải mờ là quá trình xác định giá trị rõ ở đầu ra từ hàm thuộc µB’(y) của tập mờ B’. Có 2 phương pháp giải mờ :

**Phương pháp cực đại**

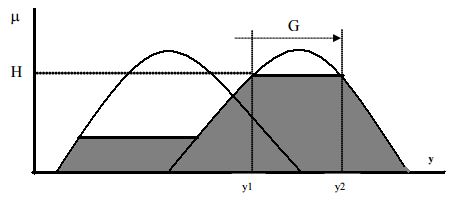
Các bước thực hiện : - Xác định miền chứa giá trị y’, y’ là giá trị mà tại đó µB’(y) đạt Max G = { y∈Y | µB’(y) = H }

- Xác định y’ theo một trong 3 cách sau :

+ Nguyên lý trung bình

+ Nguyên lý cận trái

+ Nguyên lý cận phải



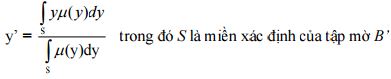
• Nguyên lý trung bình: y’ = 2 y1+ y2

• Nguyên lý cận trái : chọn y’ = y1

• Nguyên lý cận phải : chọn y’ = y2  
**Phương pháp trọng tâm**

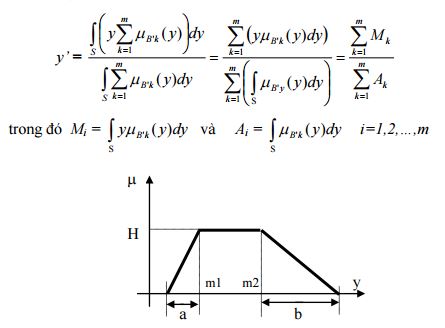
Điểm y’ được xác định là hoành độ của điểm trọng tâm miền được bao bởi trục hoành và đường µB’(y).

Công thức xác định :

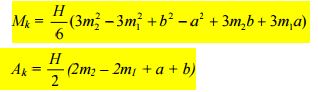


♦Phương pháp trọng tâm cho luật Sum-Min

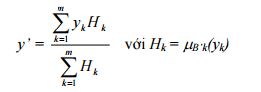
Giả sử có m luật điều khiển được triển khai, ký hiệu các giá trị mờ đầu ra của luật điều khiển thứ k là µB’k(y) thì với quy tắc Sum-Min hàm thuộc sẽ là µB’(y) =), và y’ được xác định :



Xét riêng cho trường hợp các hàm thuộc dạng hình thang như hình trên :



♦ Phương pháp độ cao



1. Các yêu cầu với bộ điều khiển mờ trong điều khiển xe tự lái

* Bộ điều khiển mờ trong xe tự lái phải giúp xe xác định rõ vật cản, tránh được vật cản tốt và đi đúng hướng được xác định.
* Bộ điều khiển phải nắm được tương đối các trường hợp vật cản để có thể xử lí trong các trường hợp đó, đưa ra vận tốc phù hợp và hướng đi để xe tránh được.
* Kết quả sau khi giải mờ là vận tốc đến đích, góc rẽ trái, góc rẽ phải (RL, RR)

1. Xây dựng tập luật mờ

Có những tập luật mờ:

+ Vận tốc:

typedef struct {

float rawValue;

float FastF;

float AvgF;

float SlowF;

float MaxBound;

float AvgBound;

float MinBound;

}VelocityFuzzy;

VelocityFuzzy velocity\_fuzzy;

+ Khoảng cách phía trước:

typedef struct {

float rawValue;

float ShortF;

float AvgF;

float LongF;

float ShortBound;

float AvgBound;

float LongBound;

}FrontFuzzy;

FrontFuzzy front\_fuzzy;

+ Khoảng cách bên trái:

typedef struct {

float rawValue;

float ShortF;

float AvgF;

float LongF;

float ShortBound;

float AvgBound;

float LongBound;

}LeftFuzzy;

LeftFuzzy left\_fuzzy;

+ Khoảng cách bên phải:

typedef struct RightFuzzy{

float rawValue;

float ShortF;

float AvgF;

float LongF;

float ShortBound;

float AvgBound;

float LongBound;

}RightFuzzy;

RightFuzzy right\_fuzzy;

+ Hướng:

typedef struct {

float rawValue;

float LeftF;

float AvgLeftF;

float MidF;

float AvgRightF;

float RightF;

float LeftBound;

float AvgLeftBound;

float AvgRightBound;

float RightBound;

}RotationFuzzy;

RotationFuzzy rotation\_fuzzy;

Sau khi có đầy đủ các thành phần trong tập luật mờ, ta tiến hành giải mờ với từng trường hợp:

+ Khoảng cách phía trước:

void fuzzy(){

if (front\_fuzzy.rawValue <= front\_fuzzy.ShortBound){

front\_fuzzy.ShortF = 1;

front\_fuzzy.AvgF = 0;

front\_fuzzy.LongF = 0;

} else if (front\_fuzzy.rawValue >= front\_fuzzy.LongBound){

front\_fuzzy.ShortF = 0;

front\_fuzzy.AvgF = 0;

front\_fuzzy.LongF = 1;

} else if (front\_fuzzy.ShortBound<front\_fuzzy.rawValue&&front\_fuzzy.rawValue<front\_fuzzy.AvgBound){

front\_fuzzy.ShortF = (front\_fuzzy.AvgBound - front\_fuzzy.rawValue) / (front\_fuzzy.AvgBound - front\_fuzzy.ShortBound);

front\_fuzzy.AvgF = (front\_fuzzy.rawValue - front\_fuzzy.ShortBound) / (front\_fuzzy.AvgBound - front\_fuzzy.ShortBound);

front\_fuzzy.LongF = 0;

} else if (front\_fuzzy.AvgBound <= front\_fuzzy.rawValue && front\_fuzzy.rawValue < front\_fuzzy.LongBound){

front\_fuzzy.AvgF = (front\_fuzzy.LongBound - front\_fuzzy.rawValue) / (front\_fuzzy.LongBound - front\_fuzzy.AvgBound);

front\_fuzzy.LongF = (front\_fuzzy.rawValue - front\_fuzzy.AvgBound) / (front\_fuzzy.LongBound - front\_fuzzy.AvgBound);

front\_fuzzy.ShortF = 0;

}

}

+Khoảng cách bên trái:

void fuzzy\_left(){

if (left\_fuzzy.rawValue <= left\_fuzzy.ShortBound){

left\_fuzzy.ShortF = 1;

left\_fuzzy.AvgF = 0;

left\_fuzzy.LongF = 0;

} else if (left\_fuzzy.rawValue >= left\_fuzzy.LongBound){

left\_fuzzy.ShortF = 0;

left\_fuzzy.AvgF = 0;

left\_fuzzy.LongF = 1;

}else if (left\_fuzzy.ShortBound < left\_fuzzy.rawValue && left\_fuzzy.rawValue < left\_fuzzy.AvgBound){

left\_fuzzy.ShortF = (left\_fuzzy.AvgBound - left\_fuzzy.rawValue) / (left\_fuzzy.AvgBound - left\_fuzzy.ShortBound);

left\_fuzzy.AvgF = (left\_fuzzy.rawValue - left\_fuzzy.ShortBound) / (left\_fuzzy.AvgBound - left\_fuzzy.ShortBound);

left\_fuzzy.LongF = 0;

}else if (left\_fuzzy.AvgBound <= left\_fuzzy.rawValue && left\_fuzzy.rawValue < left\_fuzzy.LongBound){

left\_fuzzy.AvgF = (left\_fuzzy.LongBound- left\_fuzzy.rawValue) / (left\_fuzzy.LongBound - left\_fuzzy.AvgBound);

left\_fuzzy.LongF = (left\_fuzzy.rawValue - left\_fuzzy.AvgBound) / (left\_fuzzy.LongBound - left\_fuzzy.AvgBound);

left\_fuzzy.ShortF = 0;

}

}

+Khoảng cách bên phải:

void fuzzy\_right(){

if (right\_fuzzy.rawValue <= right\_fuzzy.ShortBound){

right\_fuzzy.ShortF = 1;

right\_fuzzy.AvgF = 0;

right\_fuzzy.LongF = 0;

} else if (right\_fuzzy.rawValue >= right\_fuzzy.LongBound){

right\_fuzzy.ShortF = 0;

right\_fuzzy.AvgF = 0;

right\_fuzzy.LongF = 1;

}else if (right\_fuzzy.ShortBound < right\_fuzzy.rawValue && right\_fuzzy.rawValue < right\_fuzzy.AvgBound){

right\_fuzzy.ShortF = (right\_fuzzy.AvgBound - right\_fuzzy.rawValue) / (right\_fuzzy.AvgBound - right\_fuzzy.ShortBound);

right\_fuzzy.AvgF = (right\_fuzzy.rawValue - right\_fuzzy.ShortBound) / (right\_fuzzy.AvgBound - right\_fuzzy.ShortBound);

right\_fuzzy.LongF = 0;

}else if (right\_fuzzy.AvgBound <= right\_fuzzy.rawValue && right\_fuzzy.rawValue < right\_fuzzy.LongBound){

right\_fuzzy.AvgF = (right\_fuzzy.LongBound- right\_fuzzy.rawValue) / (right\_fuzzy.LongBound - right\_fuzzy.AvgBound);

right\_fuzzy.LongF = (right\_fuzzy.rawValue - right\_fuzzy.AvgBound) / (right\_fuzzy.LongBound - right\_fuzzy.AvgBound);

right\_fuzzy.ShortF = 0;

}

}

Sau khi có các tham số đầu vào từ các sensor, cảm biến đo được ngoài thực tế thì ta sẽ thực hiện việc giải mờ với khoảng cách phía trước, khoảng cách bên trái, khoảng cách bên phải. Tiếp tục lấy những giá trị mờ vừa được giải để chọn luật mờ tương ứng, đưa ra hướng và vận tốc phù hợp cho xe.

Ta lựa chọn phương pháp Max-Min để hợp thành các luật mờ tương ứng.

Giải mờ đối với vận tốc:

float rule\_for\_vout(FrontFuzzy f,RightFuzzy r,LeftFuzzy l,VelocityFuzzy v){

//VelocityFuzzy Vout;

float outputVout\_Slow[2];

float outputVout\_Fast[1];

float outputVout\_Avg[2];

//f=Short&l=Short&r=Short->Vout=Slow

float inputRule1[3];

inputRule1[0]=f.ShortF;

inputRule1[1]= l.ShortF;

inputRule1[2]= r.ShortF;

outputVout\_Slow[0]=getMin(inputRule1,3);

//f=Long->Vout=Fast f xa -> v nhanh

float inputRule2[1];

inputRule2[0] = f.LongF;

outputVout\_Fast[0] = getMin(inputRule2, 1);

//f=Avg->Vout=Avg

float inputRule3[1];

inputRule3[0] = f.AvgF;

outputVout\_Avg[0] = getMin(inputRule3, 1);

//f=Short->Vout=short-> v cham

float inputRule4[1];

inputRule4[0] = f.ShortF;

outputVout\_Slow[1] = getMin(inputRule4, 1);

v.SlowF=getMax(outputVout\_Slow, 2);

v.FastF =getMax(outputVout\_Fast,1);

v.AvgF = getMax(outputVout\_Avg, 2);

defuzzy\_velocity();

}

Giải mờ đối với hướng:

float ruleForRotation(FrontFuzzy f, LeftFuzzy l, RightFuzzy r, VelocityFuzzy v, DirectionTarget target){

float outRotation\_AvgLeft[2];//2 rule

float outRotation\_AvgRight[2];//2 rule

float outRotation\_Left[7];//2 rule

float outRotation\_Right[7];//2 rule

float outRotation\_Mid[2];//2 rule

//4 luat de dam bao xe luon huong ve dich

//D=Left&L=Long&&F=Long->Rotation=Left

float inputRule1[3];

inputRule1[0] = target.LeftF; inputRule1[1] = l.LongF; inputRule1[2] = f.LongF;

outRotation\_Left[0] = getMin(inputRule1, 3);

//D=Right&R=Long&F=Long->Rotation=Right

float inputRule2[3];

inputRule2[0] = target.RightF; inputRule2[1] = r.LongF; inputRule2[2] = f.LongF;

outRotation\_Right[0] = getMin(inputRule2, 3);

//D=Left&L=AvgL&F=Long->Rotation=AvgLeft

float inputRule11[3];

inputRule11[0] = target.LeftF; inputRule11[1] = l.AvgF; inputRule11[2] = f.LongF;

outRotation\_AvgLeft[0] = getMin(inputRule11, 3);

//D=Right&R=AvgR&F=Long->Rotation=AvgRight

float inputRule21[3];

inputRule21[0] = target.RightF; inputRule21[1] = r.AvgF; inputRule21[2] = f.LongF;

outRotation\_AvgRight[0] = getMin(inputRule21, 3);

//4 luat de dam bao xe luon re dung huong khi gap vat can

//f=Avg&D=Left&L=Avg->Rotation=AvgLeft

float inputRule3[3];

inputRule3[0] = f.AvgF; inputRule3[1] = target.LeftF; inputRule3[2] = l.AvgF;

outRotation\_AvgLeft[1] = getMin(inputRule3, 3);

//f=Avg&D=Right&R=Avg->Rotation=AvgRight

float inputRule4[3];

inputRule4[0] = f.AvgF; inputRule4[1] = target.RightF; inputRule4[2] = r.AvgF;

outRotation\_AvgRight[1] = getMin(inputRule4, 3);

//f=Short&L=Avg&D=Left->Rotation=Left

float inputRule5[3];

inputRule5[0] = f.ShortF; inputRule5[1] = target.LeftF; inputRule5[2] = l.AvgF;

outRotation\_Left[1] = getMin(inputRule5, 3);

//f=Short&L=Long&D=Left->Rotation=Left

float inputRule51[3];

inputRule51[0] = f.ShortF; inputRule51[1] = target.LeftF; inputRule51[2] = l.LongF;

outRotation\_Left[2] = getMin(inputRule51, 3);

//f=Short&R=Avg&D=Right->Rotation=Right

float inputRule6[3];

inputRule6[0] = f.ShortF; inputRule6[1] = target.RightF; inputRule6[2] = r.AvgF;

outRotation\_Right[1] = getMin(inputRule6, 3);

//f=Short&R=Long&D=Right->Rotation=Right

float inputRule60[3];

inputRule60[0] = f.ShortF; inputRule60[1] = target.RightF; inputRule60[2] = r.LongF;

outRotation\_Right[2] = getMin(inputRule60, 3);

//luat mid

//f=Short&D=MidF&R=Long&L=Long->Rotation=Right

float inputRule65[4];

inputRule65[0] = f.ShortF; inputRule65[1] = target.MidF; inputRule65[2] = r.LongF; inputRule65[3] = l.LongF;

outRotation\_Right[5] = getMin(inputRule65, 4);

//f=Short&D=MidF&R=Long&L=Short->Rotation=Right

float inputRule61[4];

inputRule61[0] = f.ShortF; inputRule61[1] = target.MidF; inputRule61[2] = r.LongF; inputRule61[3] = l.ShortF;

outRotation\_Right[3] = getMin(inputRule61, 4);

//f=Avg&D=MidF&R=Long&L=Short->Rotation=Right

float inputRule62[4];

inputRule62[0] = f.AvgF; inputRule62[1] = target.MidF; inputRule62[2] = r.LongF; inputRule62[3] = l.ShortF;

outRotation\_Right[4] = getMin(inputRule62, 4);

//f=Short&D=MidF&L=Long&R=Short->Rotation=Left

float inputRule63[4];

inputRule63[0] = f.ShortF; inputRule63[1] = target.MidF; inputRule63[2] = l.LongF; inputRule63[3] = r.ShortF;

outRotation\_Left[3] = getMin(inputRule63, 4);

//f=Avg&D=MidF&L=Long&R=Short->Rotation=Left

float inputRule64[4];

inputRule64[0] = f.AvgF; inputRule64[1] = target.MidF; inputRule64[2] = l.LongF; inputRule64[3] = r.ShortF;

outRotation\_Left[4] = getMin(inputRule64, 4);

//f=Avg&D=MidF&L=Long&R=Avg->Rotation=Left

float inputRule66[4];

inputRule66[0] = f.AvgF; inputRule66[1] = target.MidF; inputRule66[2] = l.LongF; inputRule66[3] = r.AvgF;

outRotation\_Left[5] = getMin(inputRule66, 4);

//2 luat lam xe re khi sap va cham hai ben

//r=Short->Rotation=Left;

float inputRule7[2];

inputRule7[0] = r.ShortF;

outRotation\_Left[6] = getMin(inputRule7, 1);

//l=Short->Rotation=Right;

float inputRule8[2];

inputRule8[0] = l.ShortF;

outRotation\_Right[6] = getMin(inputRule8, 1);

rotation\_fuzzy.AvgRightF = getMax(outRotation\_AvgRight, 2);

rotation\_fuzzy.AvgLeftF = getMax(outRotation\_AvgLeft, 2);

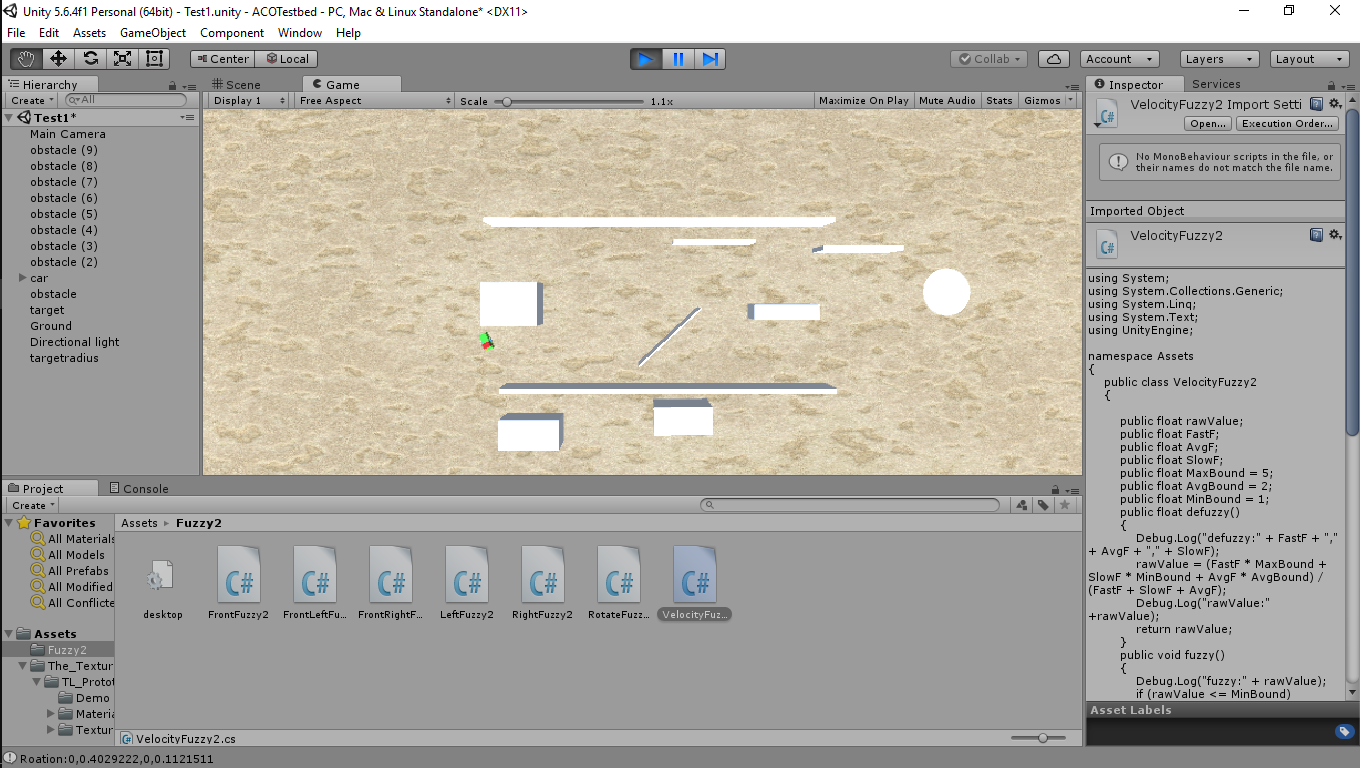
rotation\_fuzzy.RightF = getMax(outRotation\_Right, 6);

rotation\_fuzzy.LeftF = getMax(outRotation\_Left, 7);

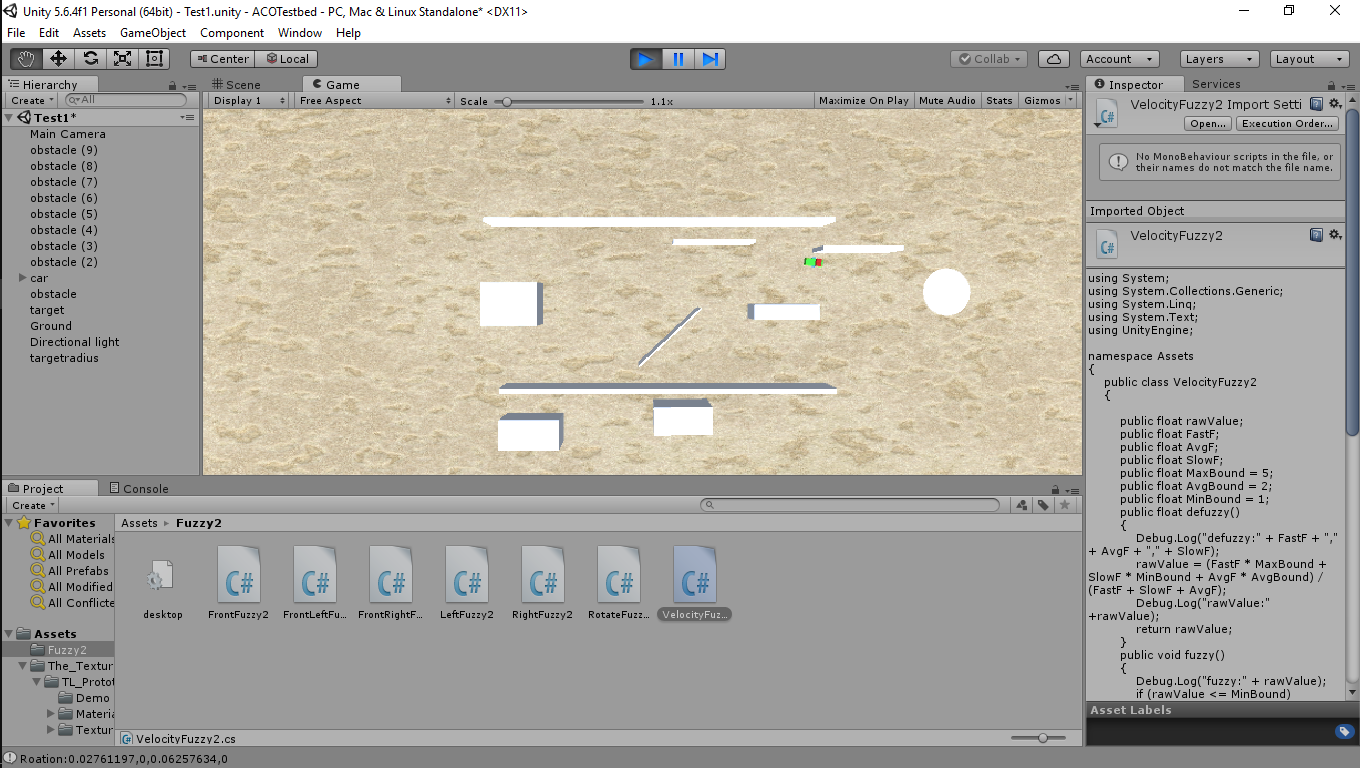
defuzzy\_rotation();

}

1. Mô phỏng bộ điều khiển mờ trên Unity

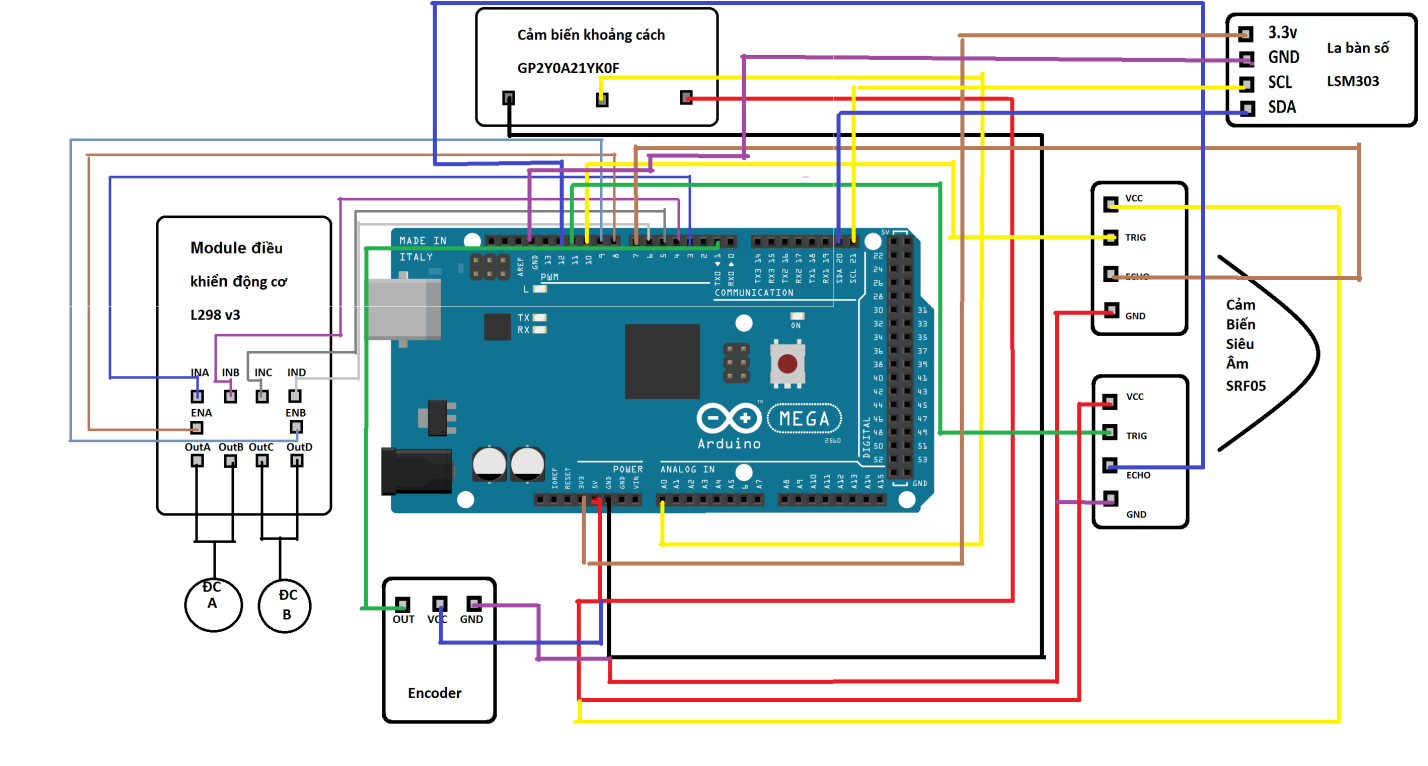


Xe ô tô đang tránh vật cản



Xe ô tô đang hướng về đích

**Chương 4 : Phát triển mô hình xe tự lái**

1. **Lắp ráp các thành phần xe tự lái**
2. Sơ đồ tổng quát :
3. Kết nối động cơ với Arduino Mega

+ Động cơ kết nối với Arduino qua 6 cổng INA, INB, INC, IND, ENA, ENB tới lần lượt các chân PWM : 3, 4, 5, 6, 8, 9.

1. Kết nối các module cảm biến khoảng cách

* Cảm biến khoảng cách hồng ngoại GP2Y0A21YK0F

+ Gồm 3 dây :

Dây đen nối với chân GND

Dây vàng nối với chân A0

Dây đỏ nối với chân 5V

* 2 cảm biến siêu âm SRF05

|  |  |  |
| --- | --- | --- |
| Cổng | Cảm biến 1 tới chân | Cảm biến 2 tới chân |
| VCC | 5V | 5V |
| Trig | 10 | 11 |
| Echo | 7 | 12 |
| GND | GND | GND |

Cách kết nối :

1. La bàn số LSM 303

La bàn số gồm 3 cổng :

Cổng 3,3V kết nối tới chân 3,3V trên arduino

Cổng GND kết nối với chân GND

Cổng SCL kết nối tới chân SCL21

Cổng SDA kết nối tới chân SDA20

1. GPS NEO-7M
2. **Cài đặt phần mềm điều khiển**
3. Cấu trúc chương trình

Sử dụng thư viện :

#include <SoftwareSerial.h>

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_LSM303\_U.h>

1. Phần thu thập dữ liệu

|  |
| --- |
| void thuKhoangCach(){ |
|  | //thu nhận dữ liệu từ sensor khoảng cách, cho ra các khoảng cách FF,FR,FL tính bằng m |
|  | unsigned long duration\_left, duration\_right; // biến đo thời gian |
|  |  |
|  | /\* Phát xung từ chân trig \*/ |
|  | digitalWrite(trig\_left,0); // tắt chân trig |
|  | delayMicroseconds(2); |
|  | digitalWrite(trig\_left,1); // phát xung từ chân trig |
|  | delayMicroseconds(5); // xung có độ dài 5 microSeconds |
|  | digitalWrite(trig\_left,0); // tắt chân trig |
|  |  |
|  | /\* Tính toán thời gian \*/ |
|  | // Đo độ rộng xung HIGH ở chân echo. |
|  | duration\_left = pulseIn(echo\_left,HIGH); |
|  | // Tính khoảng cách đến vật. |
|  | FL = (duration\_left/2/29.412/100); |
|  |  |
|  | //right |
|  | digitalWrite(trig\_right,0); |
|  | delayMicroseconds(2); |
|  | digitalWrite(trig\_right,1); |
|  | delayMicroseconds(5); |
|  | digitalWrite(trig\_right,0); |
|  | duration\_right = pulseIn(echo\_right,HIGH); |
|  | FR = (duration\_right/2/29.412/100); |
|  |  |
|  | //front |
|  | int i; |
|  | i=analogRead(redpin); |
|  | FF=((float)6762/(i-9))-4/100; |
|  | FF=(float)FF/100; |
|  | /\* In kết quả ra Serial Monitor \*/ |
|  | Serial.print("front= "); |
|  | Serial.print(FF); |
|  | Serial.println("m"); |
|  | Serial.print("left= "); |
|  | Serial.print(FL); |
|  | Serial.println("m"); |
|  | Serial.print("right= "); |
|  | Serial.print(FR); |
|  | Serial.println("m"); |
|  | delay(200); |
|  | } |
|  | float thuHuongDi(){ |
|  | //thu nhận dữ liệu từ la bàn số, tính toán hướng hiện tại D và góc lệch so với hướng đích DD |
|  | /\* Get a new sensor event \*/ |
|  | Serial.print("Compass Heading: "); |
|  | sensors\_event\_t event; |
|  | mag.getEvent(&event); |
|  |  |
|  | float Pi = 3.14159f; |
|  |  |
|  | // Calculate the angle of the vector y,x |
|  | heading = (atan2(event.magnetic.y,event.magnetic.x) \* 180) / Pi; |
|  |  |
|  | // Normalize to 0-360 |
|  | if (heading < 0) |
|  | { |
|  | heading = 360 + heading; |
|  | } |
|  | Serial.print("Compass Heading: "); |
|  | Serial.println(heading); |
|  | return heading; |
|  | //delay(500); |
|  | } |
|  | void thuVanToc(){ |
|  | //thu nhận tốc độ hiện tại V của xe bằng encoder, quy ra m/s |
|  | float n=(float)count/20; |
|  |  |
|  | //float v=1000/(millis()-lasttime)\*n; |
|  | float v=20/(100\*(millis()-lasttime))\*n; |
|  | lasttime=millis(); |
|  | Serial.println(count); |
|  | count=0; |
|  | Serial.println(v); |
|  | } |
|  | void thuToaDo(){ |
|  | //thu nhận tọa độ hiện tại, quy ra kinh độ vĩ độ Lat,Lng |
|  | } |
|  | void thuGiaTocGoc(){ |
|  | //thu nhận và tính toán gia tốc góc GR từ cảm biến gia tốc |
|  | } |
|  | void pulse() { |
|  | count++; |
|  | } |

1. Phần điều khiển

|  |  |
| --- | --- |
| void DieuKhien(){ | |
|  |
|  |
|  |
|  | if(FF>0.6){ | |
|  | //di thang | |
|  | RR=0;RL=0; Vo=0.2; | |
|  | } | |
|  | else if(FL>1){ | |
|  | RR=0;RL=1; Vo=0.1; | |
|  | } | |
|  | else if(FR>1){ | |
|  | RR=1;RL=0;Vo=0.1; | |
|  | } | |
|  | if(FF<0.15) | |
|  | Vo=-1; | |
|  | if(FF>0.6&&FR>1&&FL>1){ | |
|  |  | |
|  | } | |

1. Phần vận hành

|  |
| --- |
| void VanHanh(){ |
|  | //gồm 2 chế độ : chế độ rẽ và chế độ đi thẳng |
|  | //chế độ rẽ (nếu góc RL,RR đủ lớn) : căn cứ trên vận tốc đích Vo và các góc rẽ mà thiết lập điện áp trên 2 bánh |
|  | //chế độ đi thẳng: căn cứ trên gia tốc góc để giữ cho xe chạy thẳng với tốc độ tối đa, sử dụng PCI đơn giản |
|  | HR = RL - RR; |
|  | //Vo = 130; |
|  | int Vmin=80; |
|  | if (abs(HR) < 0.05) { |
|  | Va = Vb = V = Vo \* (255 - Vmin) + Vmin; |
|  | } |
|  | if (abs(HR) > 0.05) { |
|  | if (HR < 0) { |
|  | Va = Vo \* (225 - Vmin) + Vmin; |
|  | Vb = Va \* (1 - abs(HR)); |
|  | } |
|  | else if (HR > 0) { |
|  | Vb = Vo \* (225 - Vmin) + Vmin; |
|  | Va = Vb \* (1 - abs(HR)); |
|  | } |
|  | } |
|  | if (abs(HR) < 0.05) { |
|  |  |
|  | digitalWrite(INA, HIGH); |
|  | digitalWrite(INC, HIGH); |
|  | digitalWrite(INB, LOW); |
|  | digitalWrite(IND, LOW); |
|  | analogWrite(SPA, Va); |
|  | analogWrite(SPB, Vb); |
|  | delay(100) |
|  | ; |
|  | } |
|  | if (abs(HR) > 0.05) { |
|  |  |
|  | if (HR < 0) { |
|  | digitalWrite (INC, HIGH); |
|  | digitalWrite (INA, HIGH); |
|  | digitalWrite (INB, LOW); |
|  | digitalWrite (IND, LOW); |
|  | analogWrite(SPA, Va); |
|  | analogWrite(SPB, Vb); |
|  | delay (100); |
|  | } |
|  | else if (HR > 0) { |
|  |  |
|  | digitalWrite (INC, HIGH); |
|  | digitalWrite (INA, HIGH); |
|  | digitalWrite (INB, LOW); |
|  | digitalWrite (IND, LOW); |
|  | analogWrite(SPA, Va); |
|  | analogWrite(SPB, Vb); |
|  | delay (100); |
|  | } |
|  | } |
|  | if(Vo<0){ |
|  | digitalWrite (INC, LOW); |
|  | digitalWrite (INA, LOW); |
|  | digitalWrite (INB, HIGH); |
|  | digitalWrite (IND, HIGH); |
|  | analogWrite(SPA, Vmin); |
|  | analogWrite(SPB, Vmin); |
|  | delay (100); |
|  | } |
|  |  |
|  | Serial.print("HR= "); |
|  | Serial.println(HR); |
|  | Serial.print("Va= "); |
|  | Serial.println(Va); |
|  | Serial.print("Vb= "); |
|  | Serial.println(Vb); |
|  |  |
|  | } |

d) Các tham số của chương trình

FL: Khoảng cách bên trái

FR: Khoảng cách bên phải

FF: Khoảng cách phía trước

V: Vận tốc của xe

D: Hướng lệch hiện tại

GR: Gia tốc góc

Lat,Lng : Tọa độ

Vo: Vận tốc ban đầu của xe

Va: Vận tốc bánh A

Vb: Vận tốc bánh B

RL: Tham số quay bên phải

RR: Tham số quay bên trái

Do: Hướng đến đích

DD: Hướng lệch của xe so với hướng tới đích

Heading: Hướng lệch ban đầu so với hướng Bắc

OriginalDirection: Hướng lệch của đích so với hướng Bắc

State: Trạng thái

# Chương 5 : Kiểm thử

## Chạy kiểm thử trong phòng

* 1. Các trường hợp kiểm thử
  2. ..

## Chạy kiểm thử ngoài thực địa

Các trường hợp kiểm thử

* 1. ..

## Đánh giá kết quả

# Chương 6: Kết luận