# Chương 1: Mở đầu

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

## Arduino

* 1. Giới thiệu Arduino (vắn tắt)
  2. Arduino Mega

- Các đặc điểm của Arduino Mega

## Module điều khiển động cơ L298 v3

* 1. Giới thiệu (ngắn, vài dòng)
  2. Cách đi dây với Arduino
  3. Các lệnh điều khiển cơ bản
     1. Thiết lập các chân tương ứng với các bánh
     2. Điều khiển tốc độ của mỗi bánh

## La bàn số LSM303

* 1. Giới thiệu (Vắn tắt)
  2. Cách đi dây với Arduino
  3. Lấy dữ liệu từ LSM303
     1. Dữ liệu gia tốc góc
     2. Dữ liệu từ trường
        1. TÍnh heading từ dữ liệu từ trường

## Cảm biến khoảng cách GP2Y0A21YK0F

* 1. Giới thiệu (Vắn tắt)
  2. Cách đi dây
  3. Lấy dữ liệu

## Cảm biến siêu âm SRF05 (tương tự)

## GPS GPS NEO-7M (tương tự)

## Encoder đo vận tốc (tương tự)

# 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

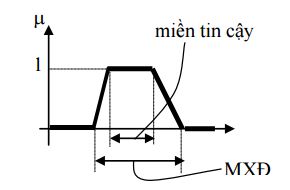
Lôgic mờ cho phép độ liên thuộc có giá trị trong khoảng đóng 0 và 1, và ở hình thức ngôn từ, các khái niệm không chính xác như "hơi hơi", "gần như", "khá là" và "rất". Cụ thể, nó cho phép quan hệ thành viên không đầy đủ giữa thành viên và tập hợp.

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.



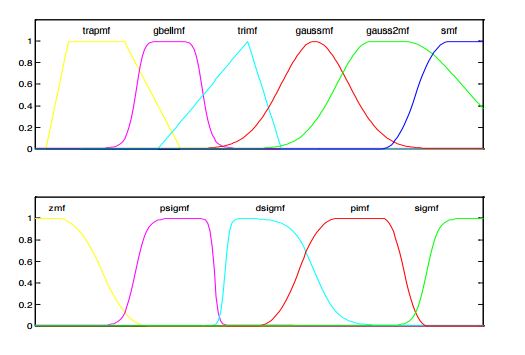
+ Các thuật ngữ trong logic mờ

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

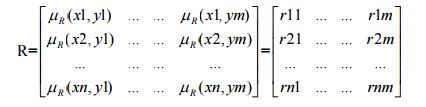
Thuật toán xây dựng mệnh đề hợp thành cho hệ SISO

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**

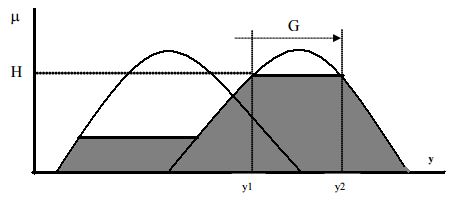
Các bước thực hiện : - Xác định miền chứa giá trị y’, y’ là giá trị mà tại đó µB’(y) đạt Max G = { y∈Y | µB’(y) = H }

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

+ Nguyên lý trung bình

+ Nguyên lý cận trái

+ Nguyên lý cận phải



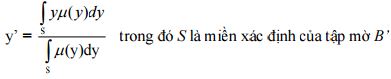
• Nguyên lý trung bình: y’ = 2 y1+ y2

• Nguyên lý cận trái : chọn y’ = y1

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

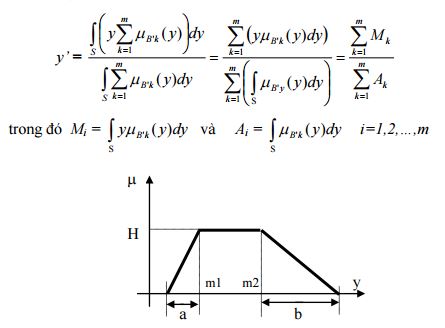
Điểm y’ được xác định là hoành độ của điểm trọng tâm miền được bao bởi trục hoành và đường µB’(y).

Công thức xác định :

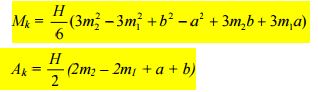


♦Phương pháp trọng tâm cho luật Sum-Min

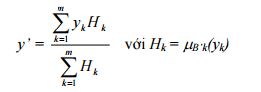
Giả sử có m luật điều khiển được triển khai, ký hiệu các giá trị mờ đầu ra của luật điều khiển thứ k là µB’k(y) thì với quy tắc Sum-Min hàm thuộc sẽ là µB’(y) =), và y’ được xác định :



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);

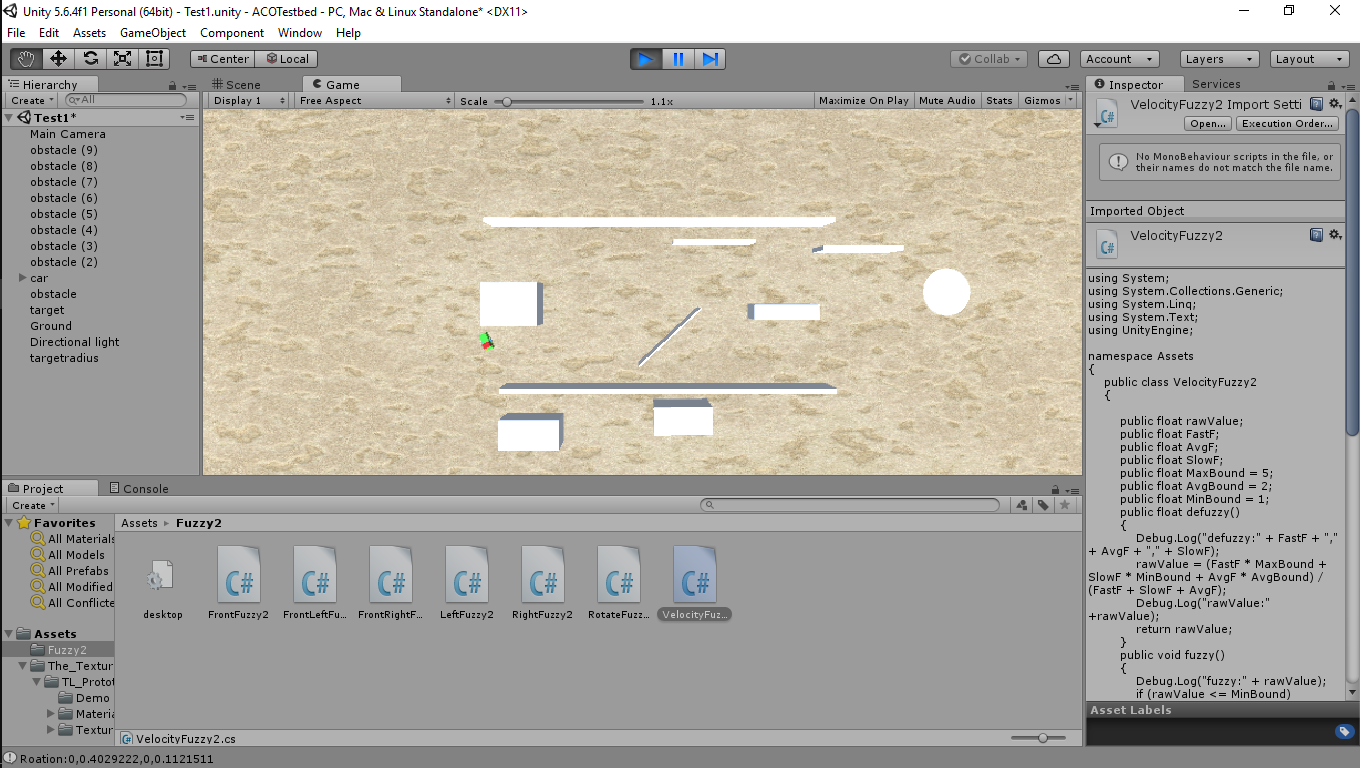
rotation\_fuzzy.RightF = getMax(outRotation\_Right, 6);

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

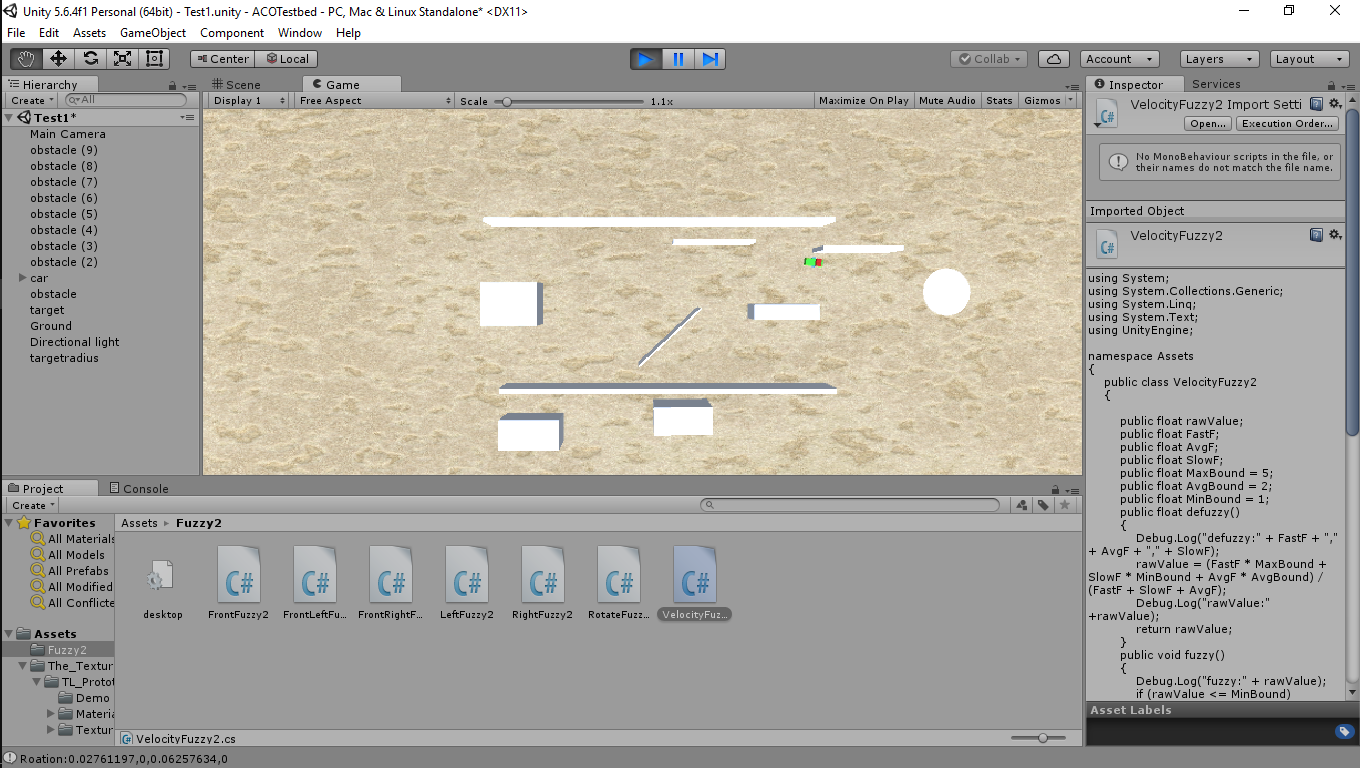
defuzzy\_rotation();

}

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

## Lắp ráp các thành phần của xe tự lái

* 1. Sơ đồ tổng quát
  2. Kết nối động cơ với Arduino Mega
  3. Kết nối các module cảm biến khoảng cách
  4. Kết nối module la bàn số
  5. Kết nối GPS Neo-7M

## Cài đặt phần mềm điều khiển

* 1. Cấu trúc chương trình
     1. Phần thu thập dữ liệu
     2. Phần điều khiển
     3. Phần vận hành
  2. Các tham số của chương trình

# Chương 4:Kiểm thử

## Chạy kiểm thử trong phòng

* 1. Các trường hợp kiểm thử
  2. ..

## Chạy kiểm thử ngoài thực địa

* 1. Các trường hợp kiểm thử
  2. ..

## Đánh giá kết quả

# Chương 5: Kết luận