BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC XÂY DỰNG**

**BÁO CÁO TỔNG KẾT**

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC SINH VIÊN NĂM 2017-2018**

**NGHIÊN CỨU VÀ PHÁT TRIỂN MÔ**

**HÌNH XE TỰ LÁI**

**Mã số đề tài:CNTT-2018-10**

**Sinh viên thực hiện, mã số sinh viên, lớp:**

**Nguyễn Mạnh Hưng-1504360-60PM2**

**Hoàng Quang Anh-1501961-61TH1**

**Đỗ Ngọc Anh-1501661-61PM2**

**Giáo viên hướng dẫn:** **ThS. Lê Đức Quang**

Mục Lục

[*CHƯƠNG 1: MỞ ĐẦU* 2](#_Toc514358989)

[*CHƯƠNG 2: GIỚI THIỆU CÁC THÀNH PHẦN PHẦN CỨNG:* 4](#_Toc514359001)

[1. Arduino 4](#_Toc514359002)

[2 Module điều khiển động cơ L298 v3 6](#_Toc514359003)

[3 La bàn số LSM303 7](#_Toc514359004)

[4 Cảm biến khoảng cách GP2Y0A21YK0F và GP2Y0A02YK0F 9](#_Toc514359005)

[5 Cảm biến siêu âm SRF05 10](#_Toc514359006)

[6 GPS GPS NEO-6M 11](#_Toc514359007)

[7 Encoder đo vận tốc 13](#_Toc514359008)

[*CHƯƠNG 3: ỨNG DỤNG ĐIỀU KHIỂN MỜ TRONG XE TỰ LÁI* 15](#_Toc514359009)

[1. Giới thiệu về điều khiển mờ 15](#_Toc514359010)

[2. Các yêu cầu với bộ điều khiển mờ trong điều khiển xe tự lái 19](#_Toc514359011)

[2.1 Xây dựng tập luật mờ 19](#_Toc514359012)

[2.2 Mô phỏng bộ điều khiển mờ trên Unity 24](#_Toc514359014)

[*CHƯƠNG 4 : PHÁT TRIỂN MÔ HÌNH XE TỰ LÁI* 25](#_Toc514359015)

[1 .Lắp ráp các thành phần xe tự lái 25](#_Toc514359016)

[2.Cài đặt phần mềm điều khiển 26](#_Toc514359017)

[*CHƯƠNG 5 : KIỂM THỬ* 32](#_Toc514359018)

[*CHƯƠNG 6: KẾT LUẬN* 36](#_Toc514359019)

# CHƯƠNG 1: MỞ ĐẦU

# 1.Đặt vấn đề

Ngày nay, công nghệ đã trở thành một phần quen thuộc với đời sống chúng ta. Từ đứa trẻ cho đến những người cao tuổi cũng có thể hiểu và sử dụng đôi chút những thiết bị điện tử. Việc ứng dụng công nghệ vào đời sống cũng không còn là một khái niệm quá xa lạ trong thời đại tự động hóa như hiện nay.

Hưởng ứng phong trào này, cùng cảm hứng được gợi từ nhu cầu sử dụng ô tô hiện tại, chúng em nghiên cứu và tìm cách sử dụng Arduino trong công nghệ chế tạo ô tô, với công nghệ này chúng ta có thế định vị một đích đến qua GPS và xe sẽ tự động di chuyển đến đích.

# 2. Mục tiêu đề tài

- Tìm hiểu các thiết bị phần cứng như Arduino & các loại cảm biến khác

- Nghiên cứu và phát triển mô hình tự ô tô tự vận hành.

# 3. Phương pháp nghiên cứu

- Nghiên cứu các thiết bị phần cứng dựa trên các hướng dẫn và tài liệu trên mạng Internet và thử nghiệm các thiết bị này để nắm được cách sử dụng cũng như các đặc điểm kỹ thuật

- Tìm hiểu các lý thuyết điều khiển, vận hành từ đó xây dựng được cách hoạt động , di chuyển của xe.

# 4. Một số mô hình hiện có:

## a) Ở Việt Nam : Mô hình xe tự lái của nhóm MTA-Racer

Là dự án mô hình xe tự lái của nhóm sinh viên Học viện Kỹ thuật Quân Sự, được sử dụng và đạt giải nhất trong cuộc đua số 2016-2017 do đại học FPT tổ chức.

Cùng cuộc thi cuộc đua số 2016-2017 còn có các mô hình của ĐH Bách Khoa, ĐH FPT,….

## b) Ở nước ngoài:

## - Dự án xe tự lái Waymo của Google:

## Waymo là dự án do bộ phận xe tự lái của Alphabet, công ty mẹ của Google phát triển.

## John Krafcik, CEO của công ty mới này, mô tả Waymo là một công ty "*với nhiệm vụ phát triển một phương tiện giao thông an toàn và dễ dàng cho tất cả mọi người*".

Trong gần như cả thập kỷ vừa qua, Google đã rót hàng núi tiền để nghiên cứu công nghệ xe tự lái. Google là hãng tiên phong trong việc đưa ra những mẫu xe hơi tự động, với một "đội quân" xe hơi mini không người lái đã lăn bánh trên những con đường tại thung lũng Silicon. Tuy nhiên, công ty mẹ của Google là Alphabet đã quyết định đại tu lại toàn bộ dự án này. Và vào năm 2016 Alphabet đã chính thức công bố rằng Waymo sẽ chịu trách nhiệm thực hiện dự án xe tự lái của họ. Không như Google, Waymo không tự mình sản xuất xe hơi mà sẽ hợp tác với các nhà sản xuất chuyên nghiệp khác.

Với công bố này của Alphabet, Waymo báo hiệu một sự chuyển hướng trong việc phát triển mạnh hơn các phương tiện của mình. Trong những năm gần đây, đầu tư vào công nghệ xe hơi tự động hoá đang bùng nổ. Các nhà sản xuất xe hơi lớn như Ford và General Motors hiện dẫn đầu trong danh sách 18 công ty có khả năng phát triển xe hơi tự lái. Waymo đứng thứ 7 trong danh sách này, và là công ty không sản xuất xe hơi có vị trí cao nhất.

## Uber cũng từng phát triển dự án xe tự lái :

Lần đầu tiên Uber công bố ý định cạnh tranh trong lĩnh vực xe tự lái vào tháng 2/2015. Tình cờ cũng chính là thời điểm Google thể hiện sự quan tâm vào công nghệ đang thu hút sự chú ý của nhiều ông lớn trong nghành công nghiệp ô tô.

Sau những thử nghiệm đầu tiên với Ford Fusion, công ty cung cấp ứng dụng gọi xe chuyển sang Volvo XC90 để phục vụ cho hoạt động nghiên cứu và phát triển công nghệ tự lái.

Uber tranh thủ sự ủng hộ và tích cực hơn trong tham vọng phát triển công nghệ tự lái, thậm chí xây dựng một thành phố "giả" có tên gọi Almono để thử nghiệm. Tuy nhiên, ngay từ giai đoạn đầu tiên đã xảy ra những vụ va chạm nhẹ và vi phạm giao thông.

## Còn một số dự án đang được phát triển bởi các hãng sản xuất ô tô nổi tiếng trên thế giới như Toyota, Ford, Volvo, Volkswagen,…

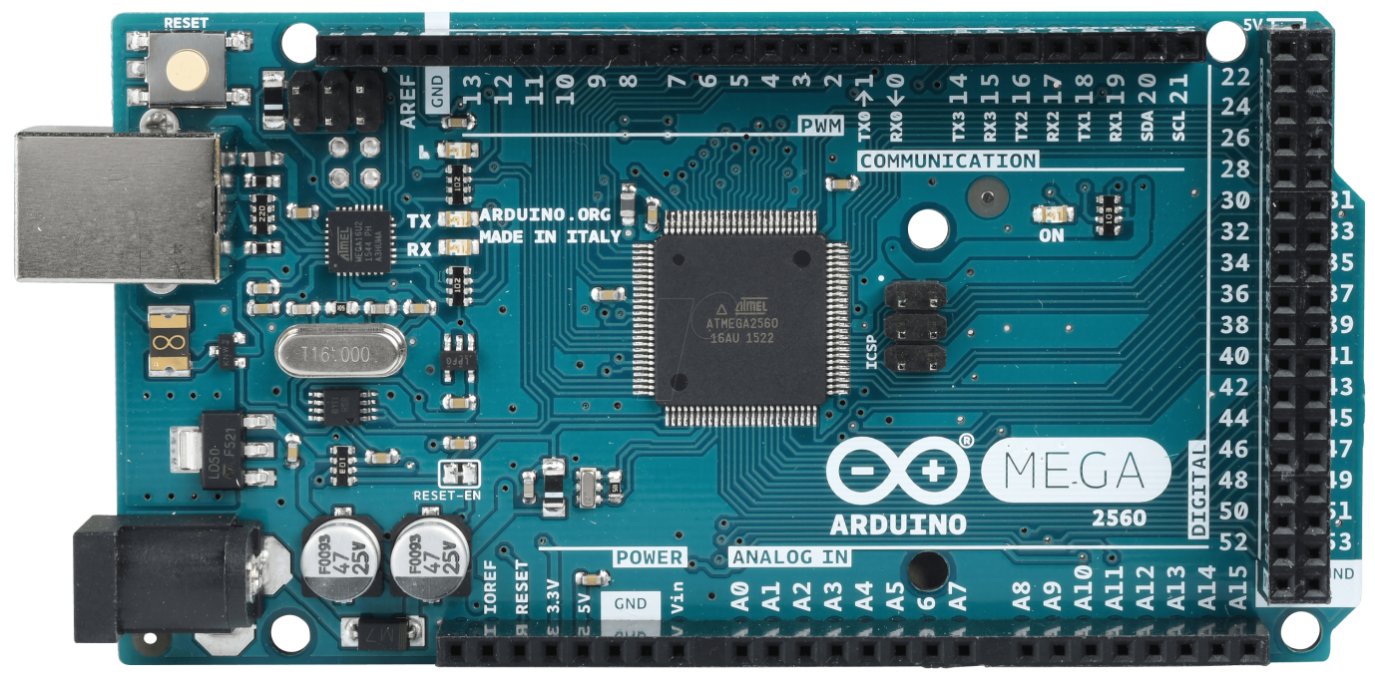
* Nhìn chung, các dự án xe tự lái cả ở trong và ngoài nước đang được phát triển một cách nhanh chóng, đánh dấu cho sự phát triển vượt bậc của nền công nghiệp tự động hóa trên thế giới

# CHƯƠNG 2: GIỚI THIỆU CÁC THÀNH PHẦN PHẦN CỨNG:

# Arduino

* 1. **Giới thiệu Arduino**
  2. **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.
  1. **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.



Hình 1: Arduino Mega2560

# 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



Hình 2: Đi dây với Arduino

* 1. Các lệnh điều khiển cơ bản

Dòng điện chạy động cơ 1 được điều khiển bởi 2 chân INA và INB, dòng điện chạy động cơ 2 được điều khiển bởi 2 chân INC và IND. Khi muốn chạy động cơ nào thì đặt điện áp mức High lên 1 chân và Low lên chân còn lại. Chân SPA và SPB để điều khiển tốc độ xung của dòng điện lên động cơ, qua đó điều chỉnh tốc độ của các động cơ này.

Trong Arduino để thiết lập điện áp lên các động cơ thông qua các chân INA,INB,INC,IND ta dùng lệnh digitalwrite

digitalWrite (INC, HIGH);

digitalWrite (INA, HIGH);

digitalWrite (INB, LOW);

digitalWrite (IND, LOW);

Để điều khiển tốc độ động cơ qua hai chân SPA và SPB

analogWrite(SPA, Va);

analogWrite(SPB, Vb);

Với Va và Vb có giá trị trong khoảng [0,255]

# 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



Hình 3: Đi dây với Arduino

* 1. Lấy dữ liệu từ LSM303

Để giao tiếp với LSM303 ta sử dụng thư viện của Adafruit

Khởi động LSM 303:

Adafruit\_LSM303\_Mag\_Unified mag = Adafruit\_LSM303\_Mag\_Unified(12345);

…

mag.enableAutoRange(true);

if(!mag.begin())

{

/\* There was a problem detecting the LSM303 ... check your connections \*/

Serial.println("Ooops, no LSM303 detected ... Check your wiring!");

while(1);

}

Để lấy thông số đo:

sensors\_event\_t event;

mag.getEvent(&event);

Lấy số đo từ trường tương ứng với 3 trục x,y,z

event.magnetic.x;

event.magnetic.y;

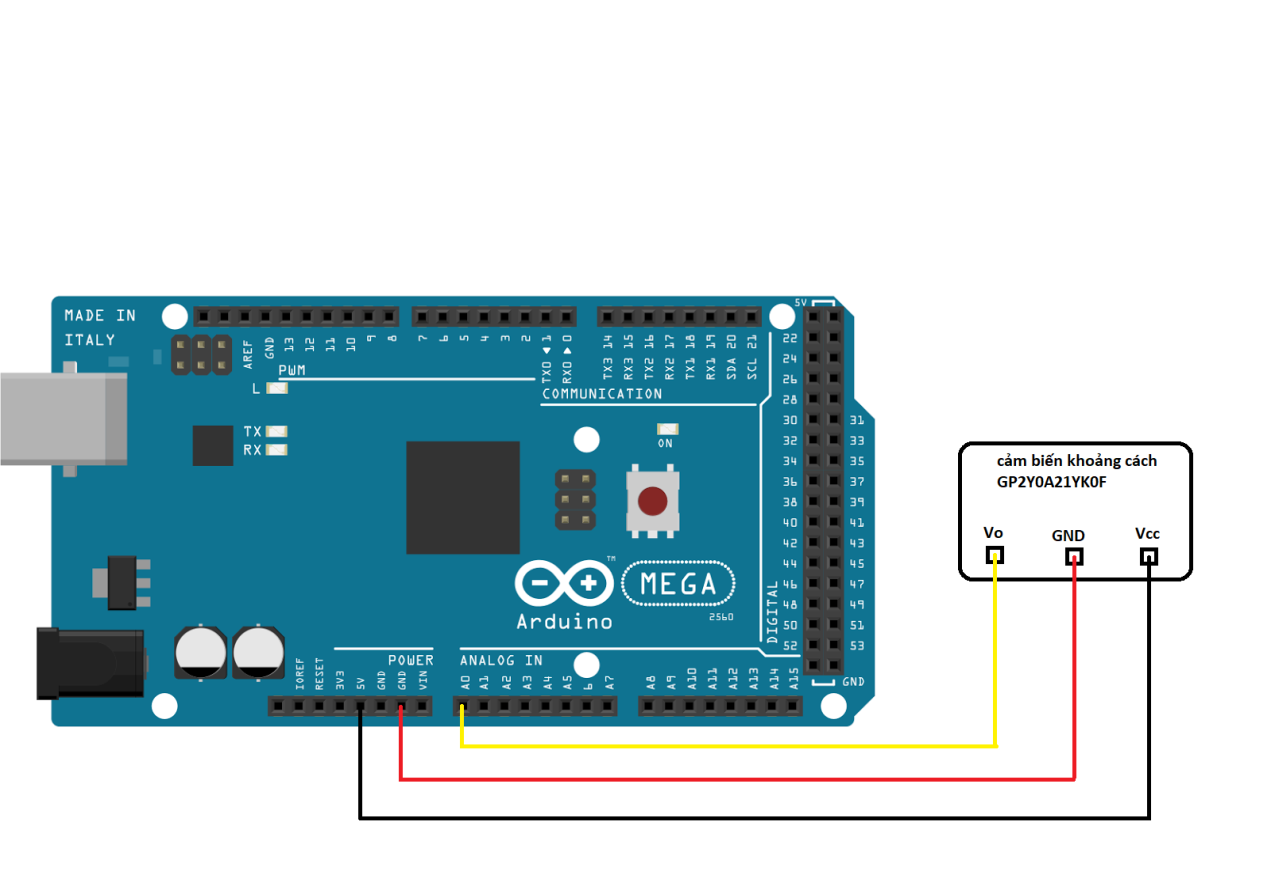
event.magnetic.z;

# Cảm biến khoảng cách GP2Y0A21YK0F và GP2Y0A02YK0F

* 1. Giới thiệu

GP2Y0A21YK0F và GP2Y0A02YK0F là cảm biến dùng tia hồng ngoại để đo khoảng cách . 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 (GP2Y0A21YK0F), 20-150 cm (GP2Y0A02YK0F)
* 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



Hình 4: Đi dây với Arduino

* 1. Lấy dữ liệu

Khai báo:

const int fpin=A0;

…

pinMode(fpin,INPUT);//fpin là chân analog kết nối với cảm biến

Tính khoảng cách đến vật:

float f;

f=analogRead(fpin);

d=(10650.08 \* pow(f,-0.935) - 10)/100;// với cảm biến GP2Y0A02YK0F

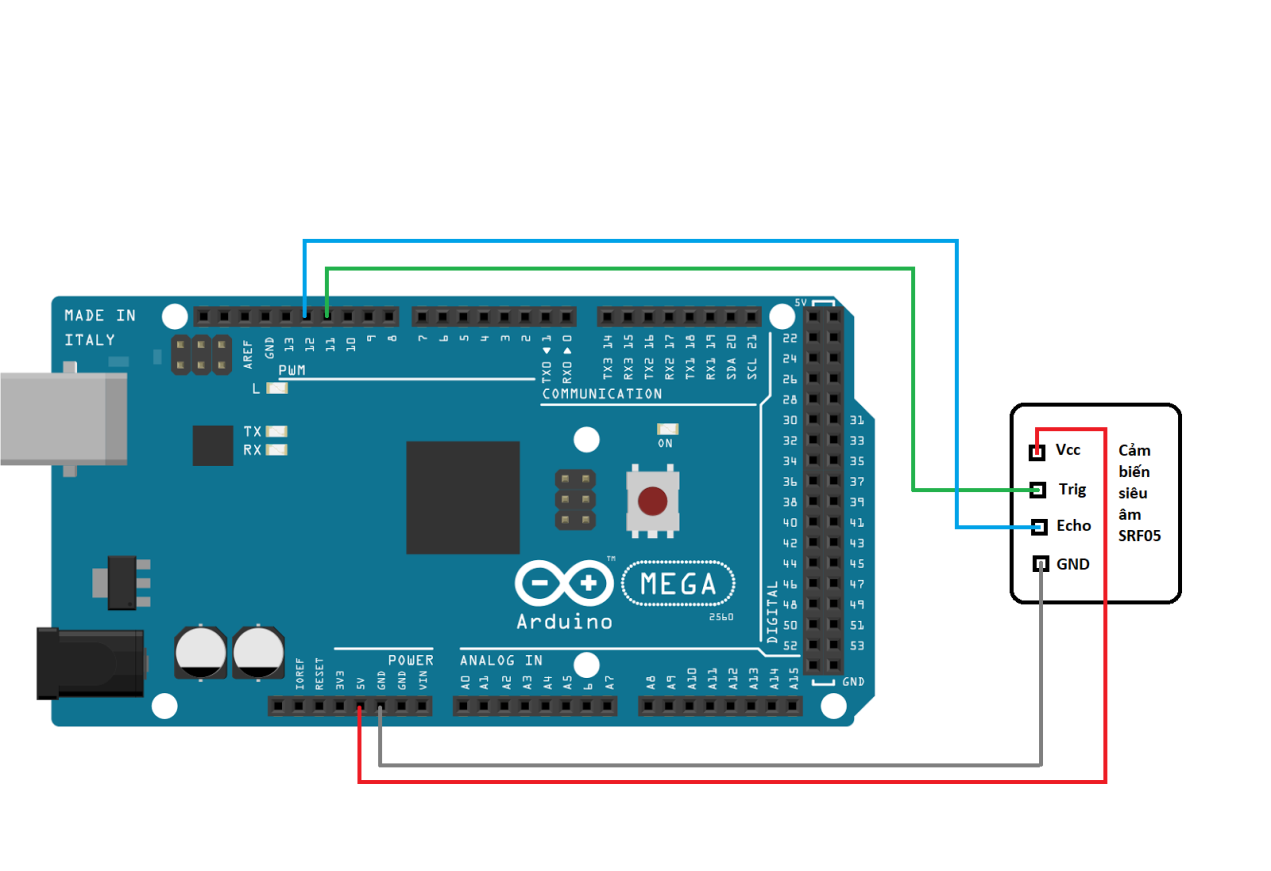
d=((6762.0/(r-9))-4)/100;// với cảm biến GP2Y0A21YK0F

# 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



Hình 5: Đi dây với Arduino

* 1. Lấy dữ liệu

Khai báo:

const int echo = 12, trig = 11;

…

pinMode(trig,OUTPUT); // chân trig sẽ phát tín hiệu

pinMode(echo,INPUT); // chân echo sẽ nhận tín hiệu

Tính khoảng cách:

unsigned long duration; // biến đo thời gian

/\* Phát xung từ chân trig \*/

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

delayMicroseconds(2);

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

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

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

/\* Tính toán thời gian \*/

// Đo độ rộng xung HIGH ở chân echo.

duration = pulseIn(echo,HIGH);

// Tính khoảng cách đến vật.

float d = (duration/2/29.412/100);

# GPS GPS NEO-6M

* 1. Giới thiệu

Với kiến trúc nhỏ gọn, mạnh mẽ và có các tùy chọn bộ nhớ đã làm cho NEO-6M lý tưởng cho các thiết bị di đông với giá rẻ và có diện tích nhỏ.

**Thông số kỹ thuật:**

* Điện áp vào: 2.7V-3.6V
* Công suất tiêu thụ: <80mW/1.8V 120mW/3.0V
* Công suất cung cấp chờ: 1.3V-4.8V, 30uA
* Kiểu Anten: active or passive
* Hỗ trợ GPS
* Hỗ trợ các chuẩn giao tiếp: UART, USB, SPI, DDC
* Hỗ trợ thạch anh RTC
* Hỗ trợ anten
* Nhiệt độ hoạt động: -40~+85°C
* Kích thước 16x12.2x2.4mm
  1. Cách đi dây với Arduino



Hình 6: cách đi dây với Adruino

* 1. Lấy dữ liệu

Để giao tiếp với GPS NEO 6M ta sử dụng thư viện: TinyGPS

Khai báo:

TinyGPS gps;

…

pinMode(18, INPUT\_PULLUP);

…

bool newData = false;

while (Serial1.available())

{

char c = Serial1.read();

// Serial.write(c); // uncomment this line if you want to see the GPS data flowing

if (gps.encode(c)) // Did a new valid sentence come in?

newData = true;

}

if (newData)

{

float flat, flon;

unsigned long age;

gps.f\_get\_position(&flat, &flon, &age);

}

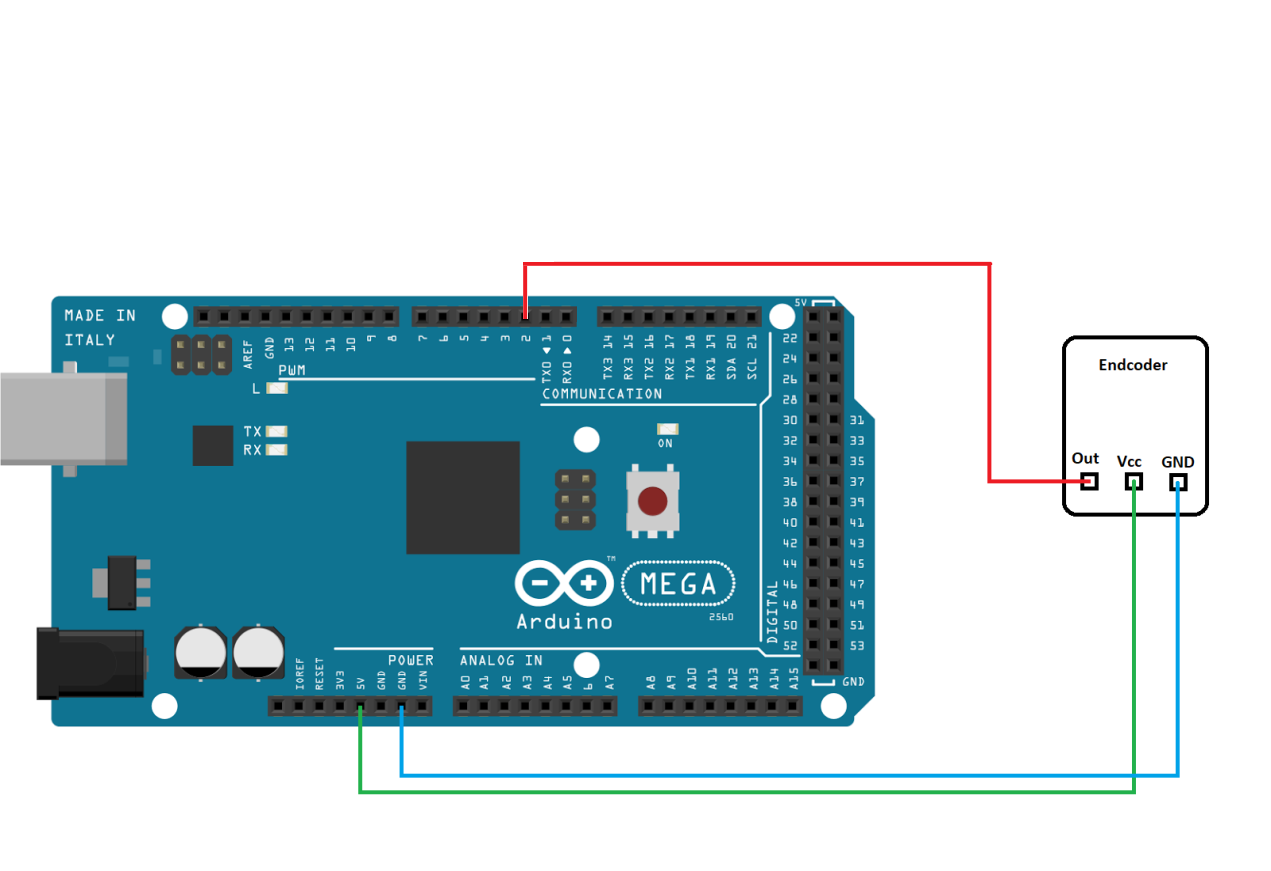
# Encoder đo vận tốc

* 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



Hình 7: Đi dây với Arduino

* 1. Lấy dữ liệu thiếu hàm pulse và lệnh ở hàm setup

Khai báo:

long lasttime;

volatile int count = 0;

…

lasttime = millis();

pinMode(2, INPUT\_PULLUP);

//attachInterrupt(0, pulse, LOW);

attachInterrupt(digitalPinToInterrupt(2), pulse, FALLING);

…

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 pulse()

{

count++;

}

# CHƯƠNG 3: ỨNG DỤNG ĐIỀU KHIỂN MỜ TRONG XE TỰ LÁI

# 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

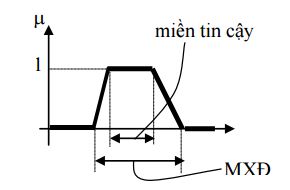
Lôgic mờ cho phép độ liên thuộc có giá trị trong khoảng đóng 0 và 1, và ở hình thức ngôn từ, các khái niệm không chính xác như "hơi hơi", "gần như", "khá là" và "rất". Cụ thể, nó cho phép quan hệ thành viên không đầy đủ giữa thành viên và tập hợp.

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.



Hình 8:

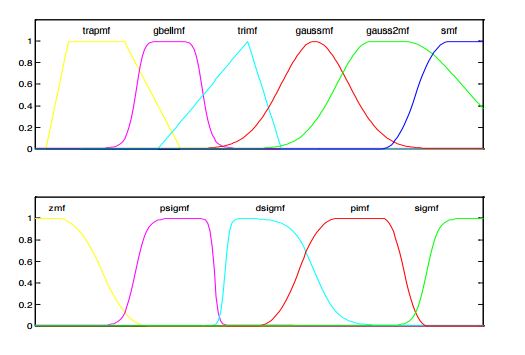
+ Các thuật ngữ trong logic mờ

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 …



Hình 9:

+ 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

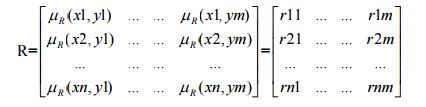
Thuật toán xây dựng mệnh đề hợp thành cho hệ SISO

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**

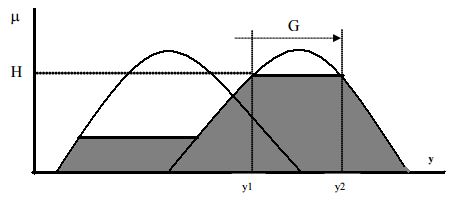
Các bước thực hiện : - Xác định miền chứa giá trị y’, y’ là giá trị mà tại đó µB’(y) đạt Max G = { y∈Y | µB’(y) = H }

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

+ Nguyên lý trung bình

+ Nguyên lý cận trái

+ Nguyên lý cận phải



Hình 10:

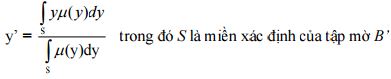
• Nguyên lý trung bình: y’ = 2 y1+ y2

• Nguyên lý cận trái : chọn y’ = y1

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

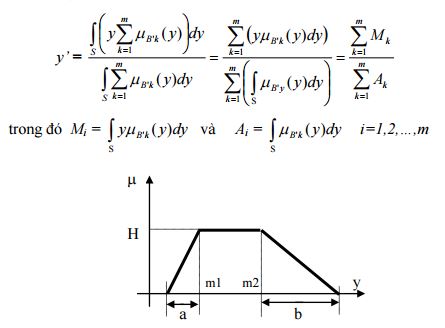
Điểm y’ được xác định là hoành độ của điểm trọng tâm miền được bao bởi trục hoành và đường µB’(y).

Công thức xác định :

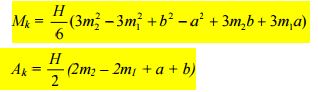


♦Phương pháp trọng tâm cho luật Sum-Min

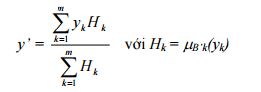
Giả sử có m luật điều khiển được triển khai, ký hiệu các giá trị mờ đầu ra của luật điều khiển thứ k là µB’k(y) thì với quy tắc Sum-Min hàm thuộc sẽ là µB’(y) =), và y’ được xác định :



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



♦ Phương pháp độ cao:



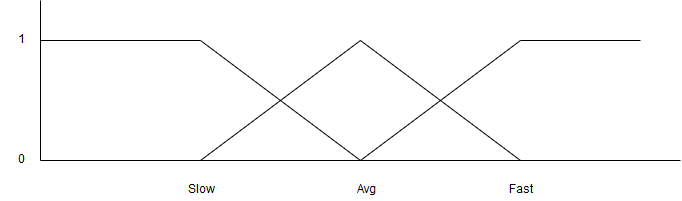
# Các yêu cầu với bộ điều khiển mờ trong điều khiển xe tự lái

* 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)

# Xây dựng tập luật mờ

# Mô tả các tập mờ tương ứng:

+ Vận tốc:



+ Khoảng cách phía trước:



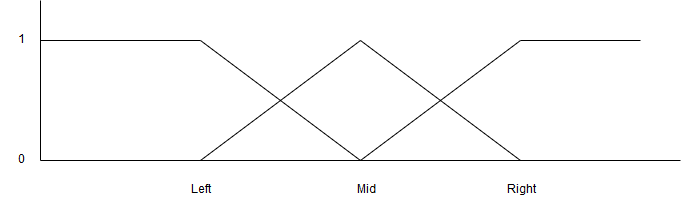
+ Khoảng cách bên trái:



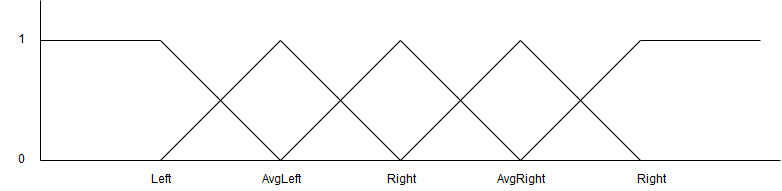
+ Khoảng cách bên phải:



+ Hướng:



+ Góc quay:



* Mờ hóa:

+Khoảng cách( trái, phải, trước):

Với các giá trị a,b,c do người dùng quyết định.

a= giá trị biên ngắn

b= giá trị biên trung bình

c=giá trị biên dài

+ Vận tốc:

Với các giá trị a,b,c do người dùng quyết định.

a= giá trị biên nhỏ nhất

b= giá trị biên trung bình

c= giá trị biên lớn nhất

+Hướng:

Với các giá trị a,b,c do người dùng quyết định.

a= giá trị biên trái

b= giá trị biên giữa

c= giá trị biên phải

* Mô tả các luật mờ:

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 góc quay và vận tốc phù hợp cho xe.

+ Luật mờ đối với VẬN TỐC:

1. Phía trước=Short & Trái=Short & Phải=Short -> Vận tốc=Slow

2. Phía trước=Long -> Vận tốc=Fast

3. Phía trước=Avg -> Vận tốc=Avg

4. Phía trước=Short -> Vận tốc=slow

+ Luật mờ đối với GÓC QUAY:

1.Đích=Left & Trái=Long & Phía trước=Long -> Quay=Left

2. Đích=Right & Phải=Long & Trước=Long -> Quay=Right

3.Đích=Left & Trái=Avg & Trước=Long->Quay=AvgLeft

4. Đích=Right & Phải=Avg & Trước=Long-> Quay=AvgRight

5.Trước=Avg & Đích=Left & Trái=Avg-> Quay=AvgLeft

6.Trước=Avg & Đích=Right & Phải=Avg-> Quay=AvgRight

7. Trước=Short & Trái=Avg & Đích=Left->Quay=Left

8.Trước=Short & Trái=Long & Đích=Left->Quay=Left

9. trước=Short&PHẢI=Avg&ĐÍCH=Right->Quay=Right

10. trước=Short&PHẢI=Long&ĐÍCH=Right->Quay=Right

11.trước=Short&ĐÍCH=MidF&PHẢI=Long&TRÁI=Long->Quay=Right

12.trước=Short&ĐÍCH=MidF&PHẢI=Long&TRÁI=Short->Quay=Right

13. trước=Avg&ĐÍCH=MidF&PHẢI=Long&TRÁI=Short->Quay=Right

14. trước=Short&ĐÍCH=MidF&TRÁI=Long&PHẢI=Short->Quay=Left

15. trước=Avg&ĐÍCH=MidF&TRÁI=Long&PHẢI=Short->Quay=Left

16. trước=Avg&ĐÍCH=MidF&TRÁI=Long&PHẢI=Avg->Quay=Left

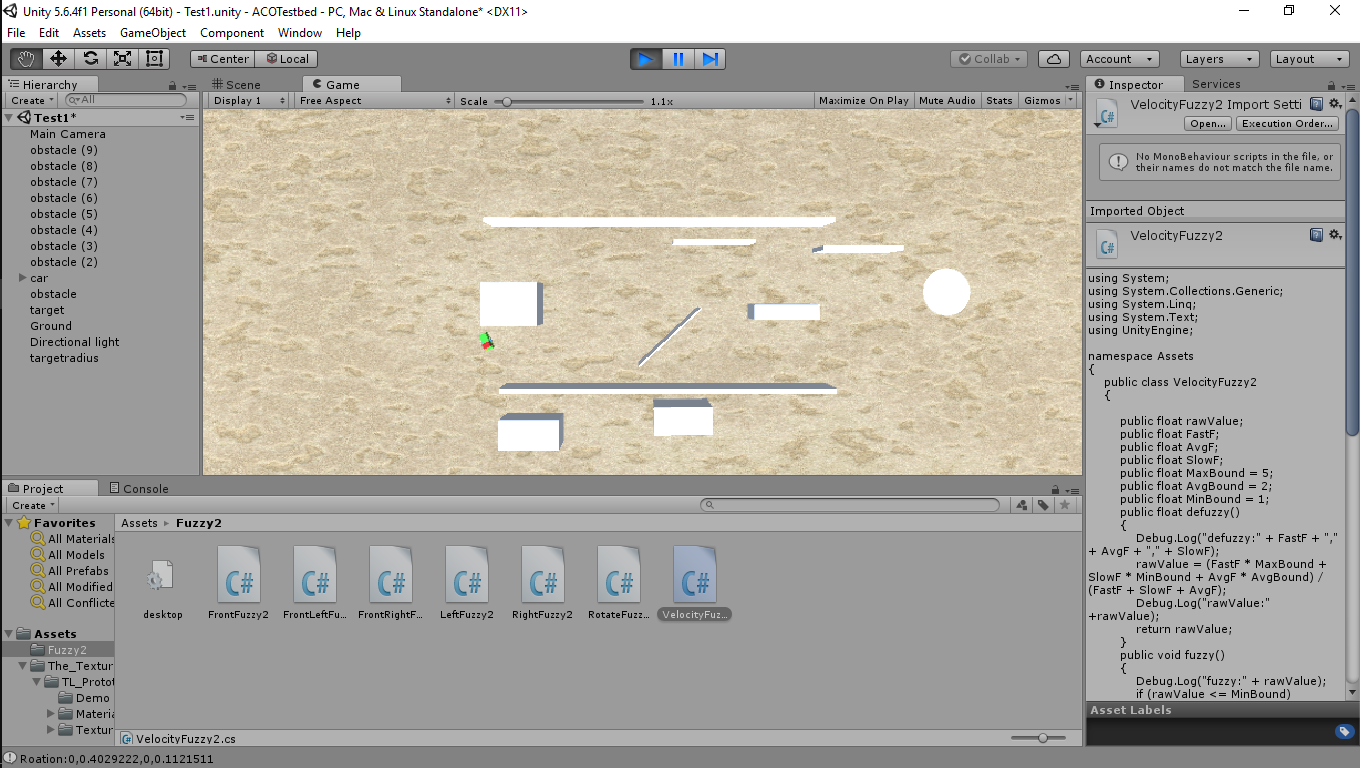
17. Phải=Short->Quay=Left

18. trái=Short->Quay=Right

* Giải mờ

Sử dụng phương pháp trọng tâm để xác định đầu ra của vận tốc và góc quay

# Mô phỏng bộ điều khiển mờ trên Unity



Xe ô tô đang tránh vật cản

# CHƯƠNG 4 : PHÁT TRIỂN MÔ HÌNH XE TỰ LÁI

# 1 .Lắp ráp các thành phần xe tự lái

1. Sơ đồ tổng quát :

***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-6M

Cổng VCC kết nối với chân 5V

Cổng GND kết nối với chân GND

Cổng RX kết nối với chân RX1

Cổng TX kết nối với chân TX1

# 2.Cài đặt phần mềm điều khiển

1. Cấu trúc chương trình

Lấy dữ liệu từ thư viện :

#include <SoftwareSerial.h>

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_LSM303\_U.h>

1. Phần thu thập dữ liệu

Quá trình thu thập dữ liệu gồm 4 quá trình thực hiện song song :

* Thu khoảng cách

Thu nhận dữ liệu từ 3 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

float i;

i=analogRead(redpin);

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

/\* In kết quả ra Serial Monitor \*/

Serial.println("m");

//delay(200);

}

* Thu hướng đi

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

{

sensors\_event\_t event;

mag.getEvent(&event);

float Pi = 3.14159;

// Calculate the angle of the vector y,x

float heading = (atan2(event.magnetic.y-31.305,event.magnetic.x-233.26) \* 180) / Pi;

//Serial.print("MagX= ");Serial.print(event.magnetic.x-233.26);Serial.print(",MagY= ");Serial.println(event.magnetic.y-31.305);

// Normalize to 0-360

//if (heading < 0)

//{

// heading = 360 + heading;

//}

//tinh chinh ve huong Bac that

heading=heading+86.75;

if(heading>180) heading=heading-360;

Serial.print("Compass Heading: ");

Serial.println(heading);

//tinh goc lech so voi goc dich

float delta=TargetHeading-heading;

Serial.print("Delta=");Serial.println(delta);

if(delta>180) delta=-(360-delta);

if(delta<-180) delta=360+delta;

DD=delta/180;

Serial.print("DD=");Serial.println(DD);

}

* Thu vận tốc

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 pulse() {

count++;

}

* Thu tọa độ

Thu nhận tọa độ hiện tại, quy ra kinh độ vĩ độ Lat,Lng:

{

bool newData = false;

while (Serial1.available())

{

char c = Serial1.read();

// Serial.write(c); // uncomment this line if you want to see the GPS data flowing

if (gps.encode(c)) // Did a new valid sentence come in?

newData = true;

}

if (newData)

{

float flat, flon;

unsigned long age;

gps.f\_get\_position(&flat, &flon, &age);

Serial.print("LAT=");

Serial.print(flat == TinyGPS::GPS\_INVALID\_F\_ANGLE ? 21.003998 : flat, 6);

Serial.print(" LON=");

Serial.print(flon == TinyGPS::GPS\_INVALID\_F\_ANGLE ? 105.841931 : flon, 6);

clat=flat ==TinyGPS::GPS\_INVALID\_F\_ANGLE ? 0.0 : flat;

clong=flon ==TinyGPS::GPS\_INVALID\_F\_ANGLE ? 0.0 : flon;

Serial.print(" SAT=");

Serial.print(gps.satellites() == TinyGPS::GPS\_INVALID\_SATELLITES ? 0 : gps.satellites());

Serial.print(" PREC=");

Serial.print(gps.hdop() == TinyGPS::GPS\_INVALID\_HDOP ? 0 : gps.hdop());

}

//tinh goc dich

float Pi = 3.14159;

float rclat=clat\*Pi/180;

float rclong=clong\*Pi/180;

float rdlat=dlat\*Pi/180;

float rdlong=dlong\*Pi/180;

// float deltalong=rclong>rdlong?(rclong-rdlong):(rdlong-rclong);

float deltalong=rdlong-rclong;

//a - c, b- d

float X=cos(rdlat)\*sin(deltalong);

float Y=cos(rclat) \* sin(rdlat)-sin(rclat) \* cos(rdlat) \* cos(deltalong);

TargetHeading=atan2(X,Y)\*180/Pi;

Serial.print(sin(deltalong));Serial.print(",");Serial.println(cos(deltalong));

Serial.print(" TargetHeading=");

Serial.println(TargetHeading);

}

1. Phần điều khiển

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) :

{

Serial.print("DD before control: ");Serial.println(DD);

Serial.print("front= ");

Serial.print(FF);

Serial.println("m");

Serial.print("left= ");

Serial.print(FL);

Serial.println("m");

Serial.print("right= ");

Serial.print(FR);

initControl(FF,FR,FL,Vo,DD);//gán rawvalue cho bên NCKH.c

calControlValue();//hàm bên NKCH.c: tính các RR,RL,Vo và gán cho cấu trúc Result

HR=getHR\_fromControl();//lấy dữ liệu từ cấu trúc Result bên NCKH.c

Vo=getVo\_fromControl();

Serial.print("HR: ");Serial.println(HR);

Serial.print("Vo: ");Serial.println(Vo);

//RL=getRL\_fromControl();

Vo=getVo\_fromControl();

if(FF<0.15)

Vo=-1;

}

1. Phần vận hành

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 :

{

int Vmin=130;

float absHR=HR<0?-HR:HR;

if (absHR < 0.05) {

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

}

if (absHR > 0.05) {

if (HR < 0) {

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

Vb = Va \* (1 - absHR);

}

else if (HR > 0) {

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

Va = Vb \* (1 - absHR);

}

}

if (abs(HR) < 0.05) {

digitalWrite(INA, HIGH);

digitalWrite(INC, HIGH);

digitalWrite(INB, LOW);

digitalWrite(IND, LOW);

analogWrite(SPA, Va);

analogWrite(SPB, Vb);

delay(200)

;

}

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 (200);

}

else if (HR > 0) {

digitalWrite (INC, HIGH);

digitalWrite (INA, HIGH);

digitalWrite (INB, LOW);

digitalWrite (IND, LOW);

analogWrite(SPA, Va);

analogWrite(SPB, Vb);

delay (200);

}

}

if(Vo<0){

digitalWrite (INC, LOW);

digitalWrite (INA, LOW);

digitalWrite (INB, HIGH);

digitalWrite (IND, HIGH);

analogWrite(SPA, Vmin);

analogWrite(SPB, Vmin);

delay (200);

}

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

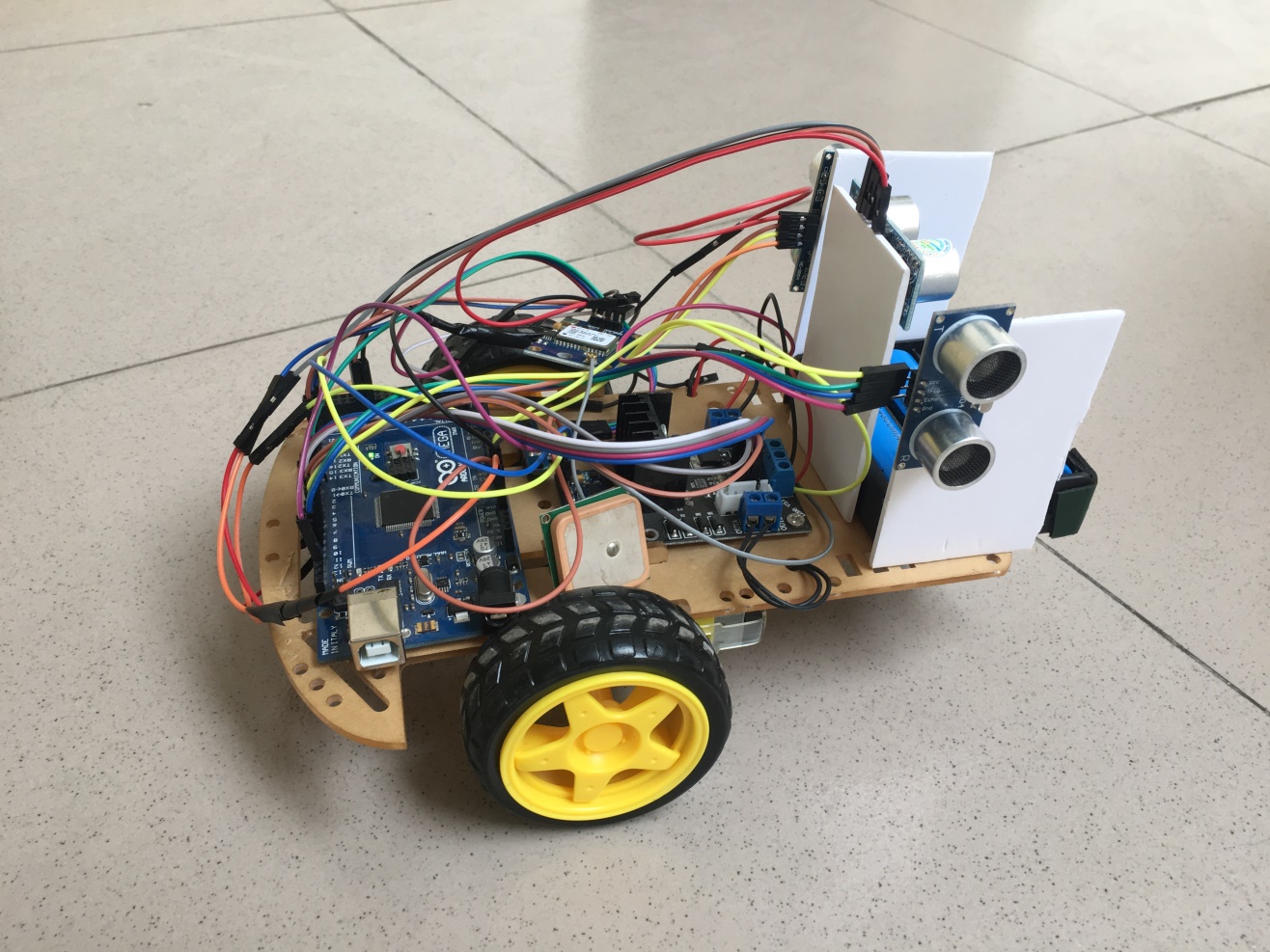
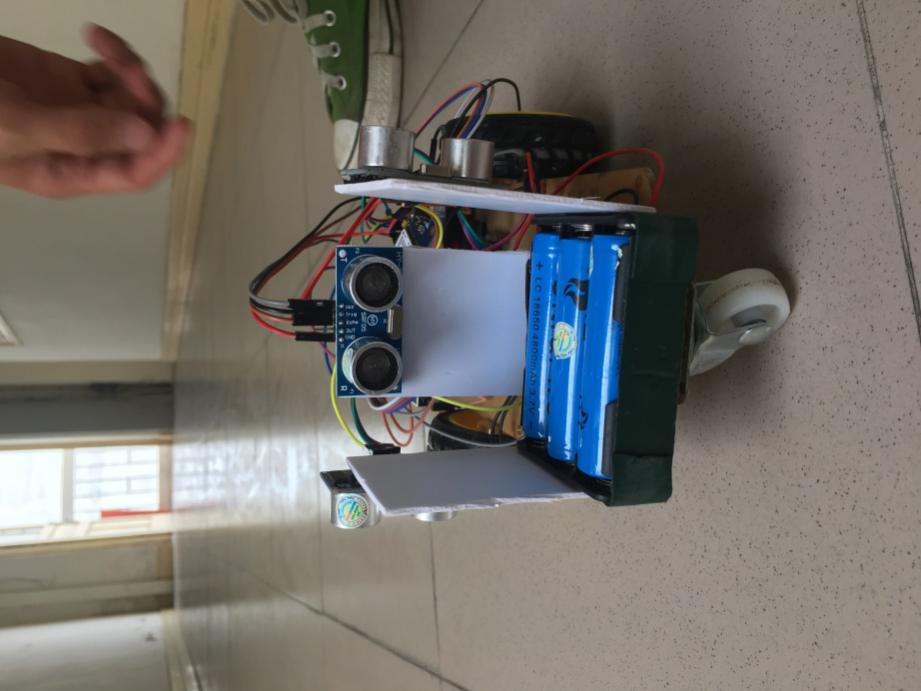
OriginalDirection: Hướng lệch của đích so với hướng Bắc

State: Trạng thái

# CHƯƠNG 5 : KIỂM THỬ

1. **Chạy kiểm thử trong phòng**
2. Nơi thử nghiệm : Trong nhà
3. Nội dung : Chạy thử tránh vật cản
4. Thiết bị :

* Module động cơ L298V3 + 2 động cơ
* Cảm biến :
* Cảm biến siêu âm SRF05
* Cảm biến khoảng cách GP2Y0A21YK0F
* La bàn số LSM303
* GPS Neo 6M
* Encoder
* Arduino Mega 2560
* Mô hình xe 3 bánh



1. Kết quả

Mô hình được chạy thử trong nhà nhiều lần, những lần đầu vẫn còn chưa nhạy bén trong quá trình cảm biển khoảng cách và góc quay nhỏ nhất của xe dẫn đến mô hình bị đâm vào tường nhưng lùi lại và tìm được hướng đi khác.

Do tốc độ cao quá nên xe chưa xử lý kịp thông tin để quay tránh vật cản trước khi vật .

Sau khi điều chỉnh các thông số thì xe có cảm nhận tốt hơn về khoảng cách, tốc độ vừa đủ để đổi hướng khi cảm nhận được vật cản.

Các thông số sau khi điều chỉnh :

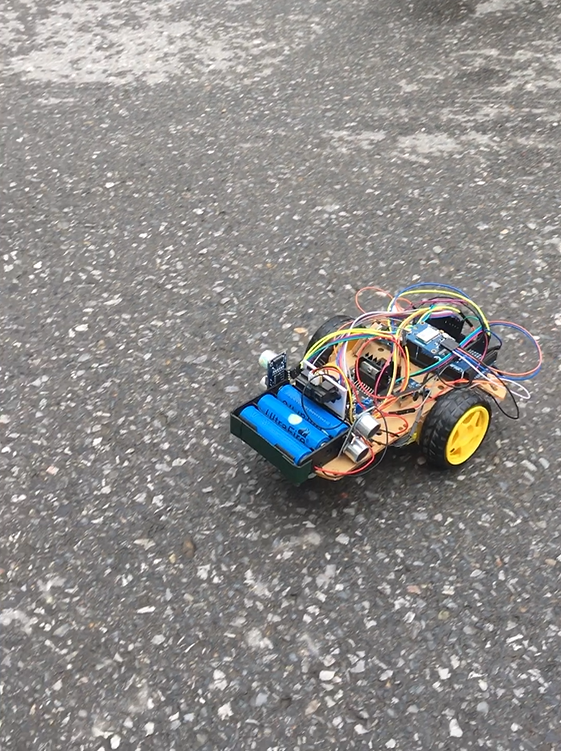
* FF( khoảng cách tới vật cản phía trước) < 0.15
* FL, FR ( khoảng cách tới vật cản bên phải và bên trái) thì cứ khi 2 module siêu âm cảm nhận được vật ở 2 bên thì xe tự động tránh
* Vmin = 80

1. **Chạy thử ngoài thực địa**
2. Nơi thử nghiệm : Ngoài trời ( chân tòa nhà A1) (clat=21.003998; clong=105.841931)
3. Nội dung : Chạy theo hướng

Đích đến : Công viên Thống Nhất (clat=21.017301; clong=105.844017)

1. Thiết bị :

* Module động cơ L298V3 + 2 động cơ
* Cảm biến :
* Cảm biến siêu âm SRF05
* Cảm biến khoảng cách GP2Y0A21YK0F
* La bàn số LSM303
* GPS Neo 6M
* Encoder
* Arduino Mega 2560
* Mô hình xe 3 bánh



1. Kết quả

Sau khi được cho chạy thử nhiều lần thì mô hình chạy đúng hướng tiến về công viên Thống Nhất.

Các thông số giữ nguyên so với chạy trong nhà.

# CHƯƠNG 6: KẾT LUẬN

**1.Kết luận**

Qua 6 tháng nghiên cứu, chúng em có kết quả như sau :

- Về lý thuyết : nghiên cứu , thử nghiệm và áp dụng các phần ứng như Arduino và các loại cảm biến ; tìm hiểu về điều khiển mờ và các ứng dụng của nó

- Về sản phẩm: Phát triển mô hình xe tự lái trong ứng dụng điều khiển mờ. Mô hình đã hoạt động cả trong nhà và ngoài trời.

**2.Phát triển**

-Tiếp tục phát triển mô hình lớn hơn, tính ứng dụng cao hơn,

-Đóng gói, hoàn thiện sản phẩm để có thể triển khai thực tế