# LỜI GIỚI THIỆU

# MỤC LỤC

# DANH SÁCH CÁC BẢNG, HÌNH VẼ

# DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

# MỞ ĐẦU

# Chương 1: TỔNG QUAN VỀ XE TỰ HÀNH

* 1. **Giới thiệu chương**

Chương này sẽ đưa ra ý tưởng, giới thiệu các chức năng đồng thời đưa ra giải pháp sơ bộ và các thành phần cấu thành hệ thống.

* 1. **Ý tưởng và yêu cầu đặt ra**
     1. ***Ý tưởng thiết kế***

Những năm gần đây, cùng với sự phát triển của khoa học công nghệ, các ngành như công nghệ thông tin và điện tử cũng phát triển mạnh mẽ hơn nhiều. Mặc dù các ngành này vốn đã phát triển mạnh từ lâu nhưng với xu hướng kết nối internet toàn cầu thì các công nghệ thông minh càng phát triển mạnh mẽ hơn.

Khi đó công nghệ tự động hóa trở nên phổ biến và thông dụng hơn trong cuộc sống con người, thay vì trước đây chủ yếu chỉ dùng trong công nghiệp, hệ thống nhúng đã chứng tỏ sự hữu dụng của nó trong các thiết bị thông minh hỗ trợ cuộc sống, như điện thoại, các thiết bị trong nhà, phương tiện di chuyển,... Bằng cách khiến chúng trở nên thông minh hơn, các thiết bị đó dễ dàng sử dụng và đảm bảo an toàn cho con người nhiều hơn.

Trong lĩnh vực ô tô, ngày càng nhiều các thiết bị đang được thay đổi từ hệ thống cơ khí sang hệ thống điện tử. Các hãng ô tô lớn trên thế giới đều cho ra mắt những mẫu xe thông minh không người lái kèm theo nhiều tiện ích hỗ trợ con người khác.

Trong khi đó, FPT Software Đà Nẵng, vốn là một công ty lớn hàng đầu về công nghệ ở Việt Nam cũng đang có những nổ lực tìm hiểu và nghiên cứu về xe tự hành, mục đích là theo kịp những công nghệ của thế giới, qua đó có thể hợp tác làm việc với những hãng xe lớn, vốn là một thị trường mà FPT Software Đà Nẵng đang hướng đến.

Những điều đó đã thôi thúc chúng tôi trong việc thực hiện đề tài “**Thiết kế và thi công mô hình xe tự hành nhận dạng và di chuyển theo làn đường**”, hệ thống sẽ phát hiện làn đường và điều khiển xe di chuyển đúng hướng.



**Hình 1.1**: Xe thông minh hiện nay có khả năng tự động tìm đường, lên lịch trình và tự động lái.

* + 1. ***Các chức năng***

Nhận dạng được làn đường.

Xác định được hướng mà xe cần di chuyển.

Điều khiển hướng di chuyển của xe cho phù hợp.

Xác định được tốc độ xe.

Kiểm soát được tốc độ xe.

* + 1. ***Các yêu cầu đặt ra***

Việc nhận dạng làn đường và xác định hướng mà xe cần di chuyển cần phải thật chính xác.

Việc điều khiển hướng di chuyển của xe phải phù hợp với hướng đã xác định được.

Việc đo tốc độ xe cần phải chính xác để có thể kiểm soát được tốc độ của xe.

* 1. **Giải pháp cho đề tài**
     1. ***Giải pháp sơ bộ***

Sử dụng Raspberry Pi 3 xử lý hình ảnh thu được từ Camera Pi với thư viện OpenCV để nhận dạng được làn đường và hướng mà xe cần di chuyển.

Gửi dữ liệu (hướng mà xe cần di chuyển) từ Raspberry PI 3 sang Arduino UNO R3 thông qua giao tiếp UART.

Điều khiển hướng di chuyển, xác định và kiểm soát tốc độ của xe bằng Arduino UNO R3.

* + 1. ***Kết quả mong đợi***

Xây dựng thành công mô hình xe tự lái có khả năng phát hiện làn đường và điều khiển xe đi đúng làn đường.

Hệ thống có tốc độ xử lý trên 20 khung hình/giây.

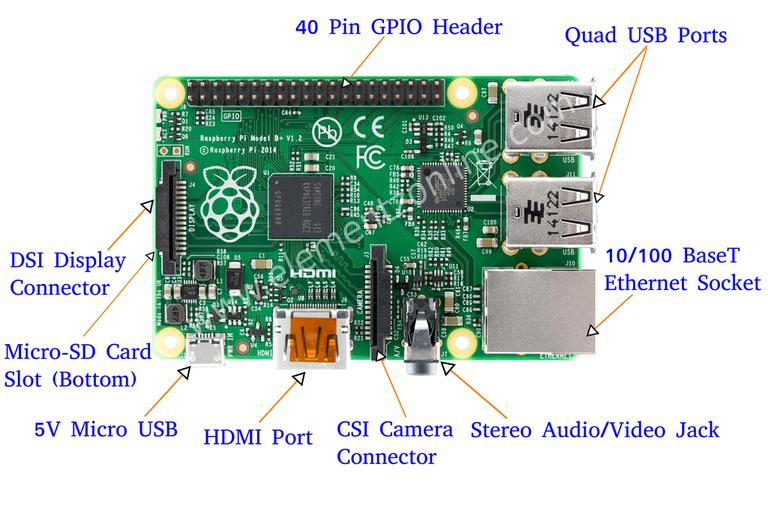


**Hình 1.2:** Xe thông minh sử dụng camera phát hiện làn đường.

* 1. **Giới thiệu các thành phần cấu thành hệ thống**
     1. ***Raspberry PI 3***

Raspberry Pi là một máy tính được thiết kế dưới dạng 1 board mạch điện tử có đầy đủ các thành phần cần thiết. Raspberry Pi sử dụng hệ điều hành linux.

Bộ xử lý trung tâm của [Raspberry Pi](http://machtudong.vn/raspberry-pi) là chip SoC (System On Chip) của Broadcom. Chip SoC tích hợp các thành phần cần thiết bao gồm: CPU, GPU, RAM trên duy nhất 1 đế chip tạo điều kiện cho việc thiết kế các hệ thống chạy ổn định nhưng lại yêu cầu kích thước nhỏ.

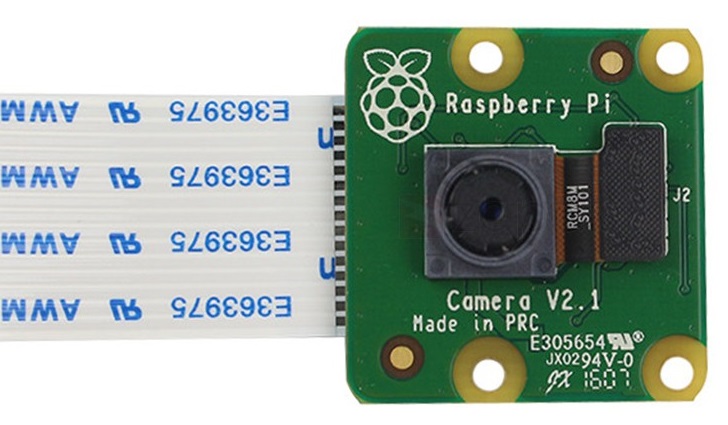


**Hình 1.3:** Mạch Raspberry pi 3

Cấu hình Raspberry Pi B+

* Chip: Broadcom BCM2835 SoC
* Core architecture: ARM11
* CPU: 700 MHz ARM1176JZFS
* GPU: Dual Core VideoCore IV® Multimedia Co-Processor. Hỗ trợ Open GL ES 2.0, tăng tốc phần cứng OpenVG, và bộ giải mã 1080p30 H.264.
* RAM: 512MB SDRAM
* Thẻ nhớ: Sử dụng MicroSD
* Kích thước 85 x 56 x 17mm
* Nguồn: Micro USB 5V, 2A
  + 1. ***Raspberry PI Camera V2***

Raspberry Pi Camera Module V2 là một bước nhảy vọt về chất lượng hình ảnh, màu sắc trung thực và hiệu suất ánh sáng thấp . Đặc biệt nó hỗ trợ video lên tới 1080P30, 720P60 và video mode VGA90, cũng như chế độ chụp hình.



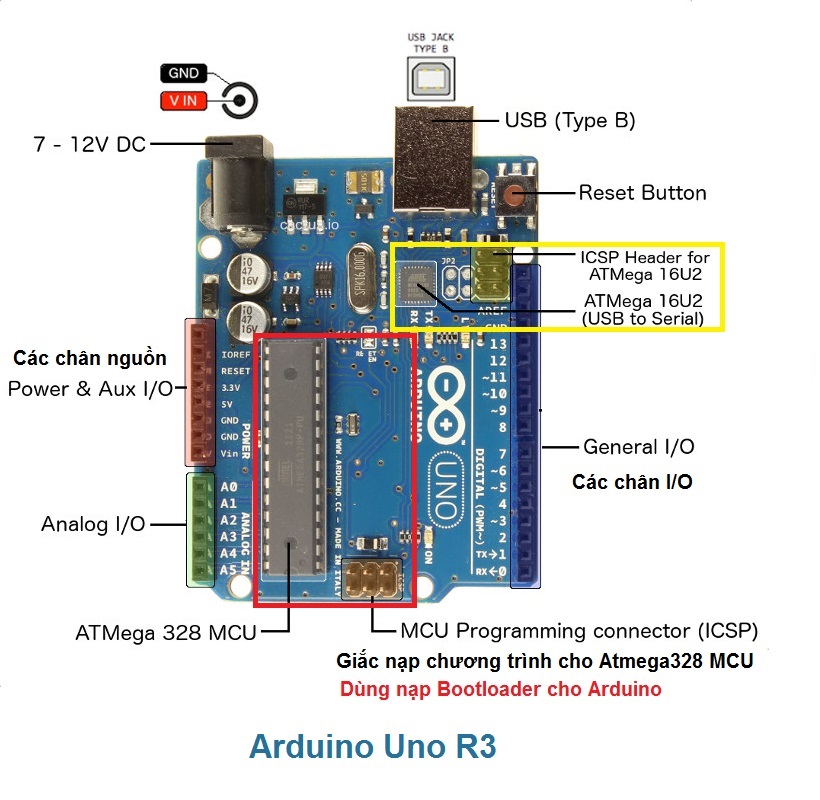
**Hình 1.5:** Module Camera PI 3

**Thông số kỹ thuật:**

* Ống kính tiêu cự cố định
* Cảm biến độ phân giải 8 megapixel cho khả năng chụp ảnh kích thước 3280 x 2464
* Hỗ trợ video 1080p30, 720p60 và 640x480p90
* Kích thước 25mm x 23mm x 9mm
* Trọng lượng chỉ hơn 3g
* Kết nối với Raspberry Pi thông qua cáp ribbon đi kèm dài 15 cm
* Camera Module được hỗ trợ với phiên bản mới nhất của Raspbian
  + 1. ***Arduino UNO R3***

***Arduino UNO R3***

Arduino board có rất nhiều phiên bản với hiệu năng và mục đích sử dụng khác nhau như: Arduino Mega, Aruino LilyPad... Trong số đó, Arduino Uno R3 là một trong những phiên bản được sử dụng rộng rãi nhất bởi chi phí và tính linh động của nó.



**Hình 1.6:** Mạch Arduino Uno R3

Arduino Uno được xây dựng với phân nhân là vi điều khiển ATmega328P sử dụng thạch anh có chu kì dao động là 16 MHz. Với vi điều khiển này, ta có tổng cộng 14 pin (ngõ) ra vào được đánh số từ 0 tới 13 (trong đó có 6 pin PWM, được đánh dấu ~ trước mã số của pin). Song song đó, ta có thêm 6 pin nhận tín hiệu analog được đánh kí hiệu từ A0 - A5, 6 pin này cũng có thể sử dụng được như các pin ra / vào bình thường (như pin 0 - 13). Ở các pin được đề cập, pin 13 là pin đặc biệt vì nối trực tiếp với LED trạng thái trên board.

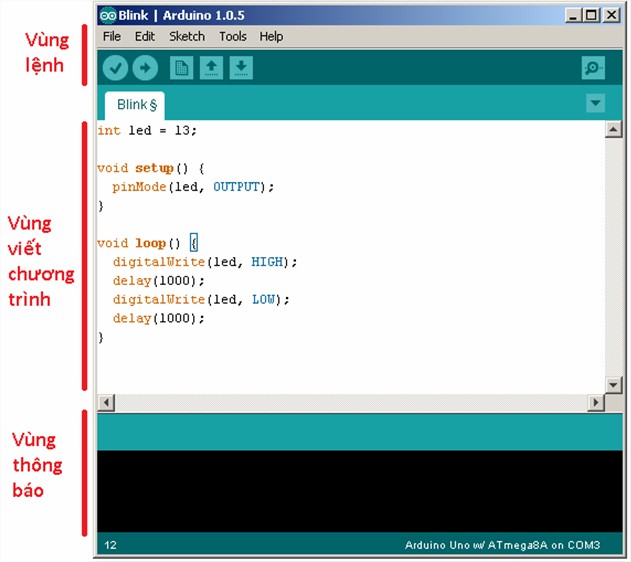
Trên board còn có 1 nút reset, 1 ngõ kết nối với máy tính qua cổng USB và 1 ngõ cấp nguồn sử dụng jack 2.1mm lấy năng lượng trực tiếp từ AC-DC adapter hay thông qua ắc-quy nguồn.

**Thông số của Arduino UNO R3:**

|  |  |
| --- | --- |
| Vi điều khiển | ATmega328 họ 8bit |
| Điện áp hoạt động | 5V DC (chỉ được cấp qua cổng USB) |
| Tần số hoạt động | 16 MHz |
| Dòng tiêu thụ | khoảng 30mA |
| Điện áp vào khuyên dùng | 7-12V DC |
| Điện áp vào giới hạn | 6-20V DC |
| Số chân Digital I/O | 14 (6 chân hardware PWM) |
| Số chân Analog | 6 (độ phân giải 10bit) |
| Dòng tối đa trên mỗi chân I/O | 30 mA |
| Dòng ra tối đa (5V) | 500 mA |
| Dòng ra tối đa (3.3V) | 50 mA |
| Bộ nhớ flash | 32 KB (ATmega328) với 0.5KB dùng bởi bootloader |
| SRAM | 2 KB (ATmega328) |
| EEPROM | 1 KB (ATmega328) |

***Arduino IDE***

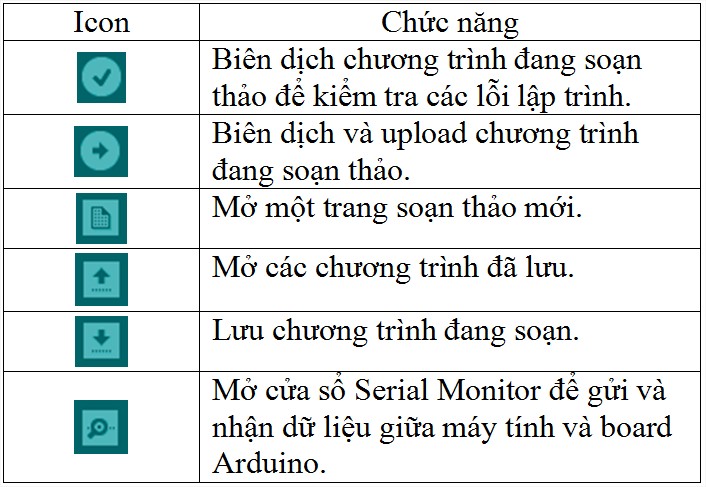
Để lập trình được cho các board Arduino, các bạn cần phải có một công cụ gọi là **I**ntergrated **D**evelopment **E**nvironment (IDE). Công cụ này được đội ngũ kĩ sư của Arduino phát triển và có thể chạy trên Windows , MAC OS X và Linux



**Hình 1.7:** Giao diện Arduino IDE

**Vùng lệnh:**

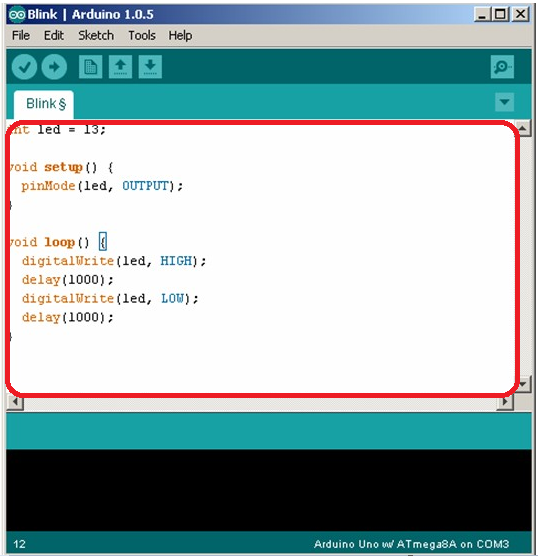
Bao gồm các nút lệnh menu (File, Edit, Sketch, Tools, Help). Phía dưới là các icon cho phép sử dụng nhanh các chức năng thường dùng của IDE được miêu tả như sau:

[](http://k3.arduino.vn/img/2014/05/24/0/465_8121-1400927938-0--table.jpg)

**Hình 1.8:** Bảng các nút lệnh trên phần mềm IDE Arduino

**Vùng viết chương trình:**

Bạn sẽ viết các đoạn mã của mình tại đây. Tên chương trình của bạn được hiển thị ngay dưới dãy các Icon, ở đây nó tên là “**Blink**”. Để ý rằng phía sau tên chương trình có một dấu “**§**”. Điều đó có nghĩa là đoạn chương trình của bạn chưa được lưu lại.



**Hình 1.9:** Vùng viết chương trình.

**Vùng thông báo (debug):**

[](http://k1.arduino.vn/img/2014/05/24/0/497_1231-1400928031-0--deb.jpg)

**Hình 1.10:** Vùng thông báo (debug)

Những thông báo từ IDE sẽ được hiển thị tại đây. Để ý rằng góc dưới cùng bên phải hiển thị loại board Arduino và cổng COM được sử dụng. Luôn chú ý tới mục này bởi nếu chọn sai loại board hoặc cổng COM, bạn sẽ không thể upload được code của mình.

* + 1. ***Servo***

Động cơ servo là loại động cơ cho phép ta điều khiển một cách cực kì chính xác. Vì vậy, khác với động cơ thông thường ta chỉ cần cấp nguồn cho động cơ là có thể vận hành được. Động cơ servo yêu cầu ta phải cấp nguồn (2 dây) và nhận điều khiển từ mạch chính (1 dây), trong đề tài này chúng tôi sử dụng servo E6001 có các dây được đánh màu như sau:

* Đỏ: nhận điện nguồn VCC.
* Đen: nối với cực âm GND của mạch
* Trắng: nhận tín hiệu từ mạch điều khiển



**Hình 1.11:** Động cơ Servo E6001

Thông số kỹ thuật: TSD Servo E6001- Servo analog 6kg

Chất liệu: bánh răng nhựa  
Thương hiệu: TSD  
Mô hình: E6001  
Kích thước: 40.5\*20.5\*50mm  
Trọng lượng: 42.4g  
Chiều dài cáp: 23cm   
Tốc độ:

0,13 giây/60 độ( 4.8V)

0,12 giây/60 độ( 6.0v)  
Mô-men xoắn:

5.2kg\*cm( 4.8V)

6.0kg\*cm( 6.0v)

Góc quay: +/- 90 độ

* + 1. ***Bộ điều tốc ESC***

ESC là thiết bị điều khiển tốc độ động cơ chổi than thường được ứng dụng rất nhiều trong các mô hình xe và máy bay với đa dạng chủng loại và công suất. Module ESC được chúng tôi sử dụng là BDESC-S10E-RTR.

Module ESC BDESC-S10E-RTR động cơ chổi than BEC.

- Pin Hỗ trợ: 2S & 3S Lipo, 4-12Cells Ni-MH & Ni-Cd.

- BEC: 5V, 2A

- Điện áp cut-off thấp, bảo vệ quá nhiệt, chống quá tải, tự động tìm vị trí trung tâm.

- Thích hợp: Xe ôtô RC và thuyền RC.



**Hình 1.12:** Bộ điều tốc ESC S10E-RTR

Thường thì ESC có 9 dây, trong đó :

Hai dây là nguồn nuôi động cơ và mạch điều khiển. Nguồn này là nguồn công suất lớn, dòng tiêu thụ có thể lên đến trên 20A tùy loại động cơ. Bạn có thể sử dụng pin lipo, hoặc nếu không mang tính di động bạn có thể sử dụng nguồn xung (nguồn tổ ong) Ampe lớn để thử động cơ.

Hai dây kết nối với động cơ.

Hai dây kết nối với công tắc để bật/tắt động cơ.

Ba dây nhận tín hiệu điều khiển sử dụng như RC servo, lưu ý ở RC servo thì ta cần cung cấp nguồn để servo hoạt động nhưng ở ESC trong 3 dây này dây có ký hiệu màu đỏ (hoặc cam) sẽ cấp nguồn 5V ngược ra môi trường, dây màu đen để nối với GND, màu trắng để nhận tns hiệu điều khiển động cơ.

* + 1. ***Động cơ***

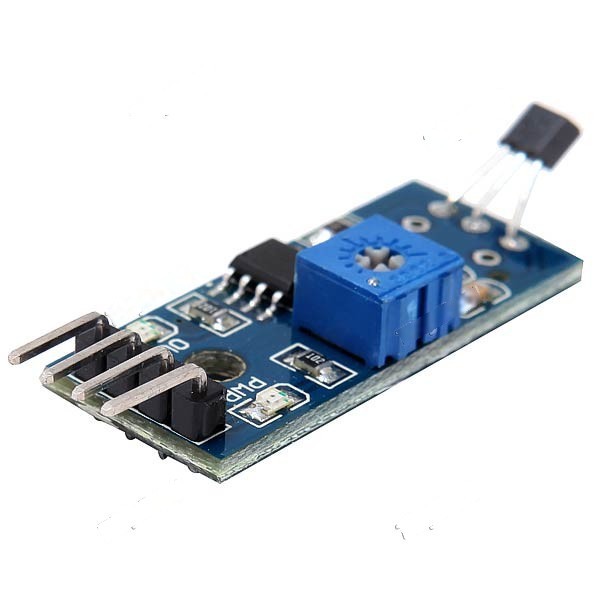
Động cơ được sử dụng ở đây là động cơ không chổi than Motor RC 540 Brushed 24.000 rmp



**Hình 1.13:** Động cơ không chổi than Motor RC 540 Brushed 24.000 rmp

* + 1. ***Cảm biến từ trường HALL***

Cảm biến từ trường được ứng dụng để phát hiện từ trường của môi trường xung quanh. Nó được ứng dụng trong bộ đo tốc độ, dùng làm mạch đếm hoặc dùng để phát hiện vị trí của vật,...

****

**Hình 1.14:** Module Cảm biến từ trường HALL

**Thông số kỹ thuật:**

* Điện áp cung cấp: 5V.
* Đèn led báo tín hiệu đầu ra.
* Sử dụng sensor 3144E và LM393.
* Dạng ngõ ra kiểu số (có thể điều chỉnh độ nhạy bằng biến trở).
* Tín hiệu tích cực ở mức thấp (0V).
* Kích thước: 20 x 32 x 11mm.
  + 1. ***Nguồn***

Hệ thống có 2 nguồn điện riêng biệt để cung cấp nguồn cho Raspberry PI 3 và động cơ, servo. Arduino UNO R3 được kết nối trực tiếp với Raspberry nên sẽ không cần nguồn riêng.

**Nguồn cho Raspberry PI 3**

Nguồn được sử dụng cho Raspberry là Module Raspberry Portable Battery. Đây là module pin rời được Raspberry sản xuất cho Raspberry để phục vụ cho việc di chuyển Raspberry.



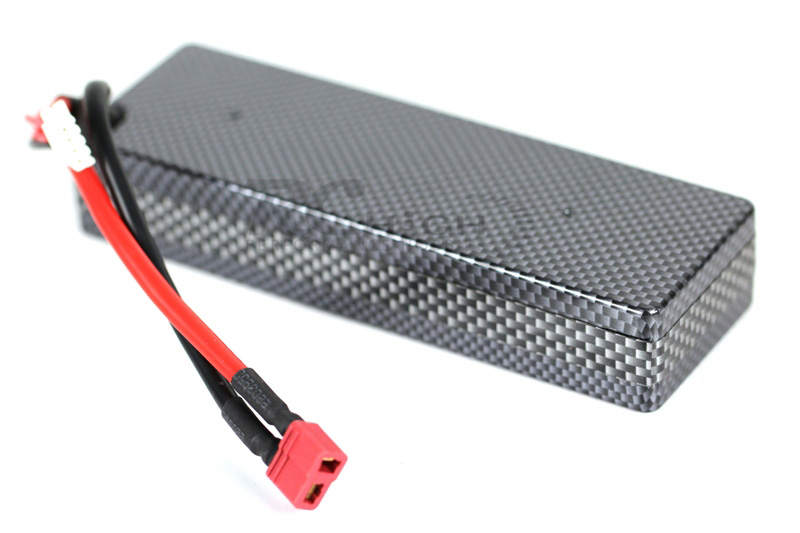
**Hình 1.15:** Module Raspberry Portable Battery

**Nguồn cho động cơ và servo**

Động cơ và servo sẽ dùng chung nguồn là pin Li-po 7.2V 3600mAh. **Pin LI- PO** có tên đầy đủ là Lithium-Ion Polymer người ta rút gọn thành Lithium-Polymer để tránh nhầm lẫn với Li-Ion. Pin LiPo không sử dụng chất điện phân dạng lỏng mà thay vào đó nó sử dụng chất điện phân dạng polymer khô, tương tự như một miếng phim nhựa mỏng. Miếng phim này được kẹp (thực sự là ghép lá) giữa cực dương và cực âm của pin cho phép trao đổi ion - do đó có tên là lithium polymer. Phương pháp này cho phép Pin có thể làm rất mỏng với các hình dạng và kích thước của cell pin khác nhau. Ưu điểm của nó:

* Pin Li-Po nhỏ, nhẹ và có thể làm ở mọi hình dáng kích thước.
* Pin  Li-Po có dung lượng cao có nghĩa là nó chứa được nhiều năng lượng trong một gói pin nhỏ.
* Pin  Li-Po có dòng xả cao để cung cấp năng lượng liên tục cho thiết bị di động

=> Đây là lý do những tại sao các nhà sản xuất lại tin dùng Pin Li-Po hơn Li-Ion. Còn một ưu điểm cũng là nhược điểm của Li-Po chưa được nhắc đến đó là loại pin này rất ít cháy nhưng khi cháy thì lại cháy lớn, và có thể phát nổ khi sạc không đúng cách hoặc sạc quá mức.

****

**Hình 1.16:** Pin Lithium dung cho động cơ và Servo

1. **Kết luận chương**

Như vậy, ở chương này ta có thể biết được sơ bộ về ý tưởng, nhiệm vụ, mục đích của đề tài “**Thiết kế và thi công mô hình xe tự hành nhận dạng và di chuyển theo làn đường**” cũng như những phương án, thiết bị được sẽ sử dụng.

# Chương 2: CƠ SỞ LÝ THUYẾT

1. **Giới thiệu chương**

Ở chương trước, ta đã có cái nhìn tổng quan về nội dung đề tài cũng như các phương án về thuật toán và thiết bị sẽ sử dụng. Chương này ta sẽ nói về những cơ sở lý luận, lý thuyết về xe tự hành, về xử lý ảnh và các thuật toán được sử dụng.

1. **Xe tự hành**

***2.2.1 Khái niệm***

Như ta đã biết, các loại xe ôtô thông thường chỉ do người lái, chỉ cung cấp 1 số phương tiện giải trí hoặc tiện ích phục vụ cho việc lái xe của người lái. Những mẫu xe thông thường hiện nay được cải tiến rất nhiều so với trước đây, tuy nhiên những cải tiến đó chủ yếu là về mặt cơ khí động lực và người lái vẫn sẽ là bộ não để điều khiển và vận hành lái xe.

Khác với các mẫu xe ôtô thông thường, xe tự hành hay còn gọi là xe tự lái hay xe thông minh được phát triển nhiều về mặt điện tử, về lập trình nhúng, chúng trở nên thông minh hơn và có 1 bộ não để suy nghĩ làm việc, cụ thể là chúng có thể lên lịch trình đường ngắn nhất, tránh kẹt xe, nhận diện đường hay vật cản để tự lái, ngoài ra các thiết bị tiện ích trên xe đều theo công nghệ IOT kết nối moi thứ trên xe 1 cách thống nhất và đơn giản. Khi đó, việc lái xe cũng giống như được thư giãn vậy.

***2.2.2 Tình hình phát triển***

Hiện nay, có rất nhiều hãng công nghệ lớn đã và đang nghiên cứu về lĩnh vực xe tự hành như Google, Tesla, Toyota, Apple… Trong số đó, có lẽ nổi bật nhất là hai mẫu xe tự hành Model S và Model X của Tesla cho phép xe tự vận hành trên nhiều kiểu đường khác nhau.

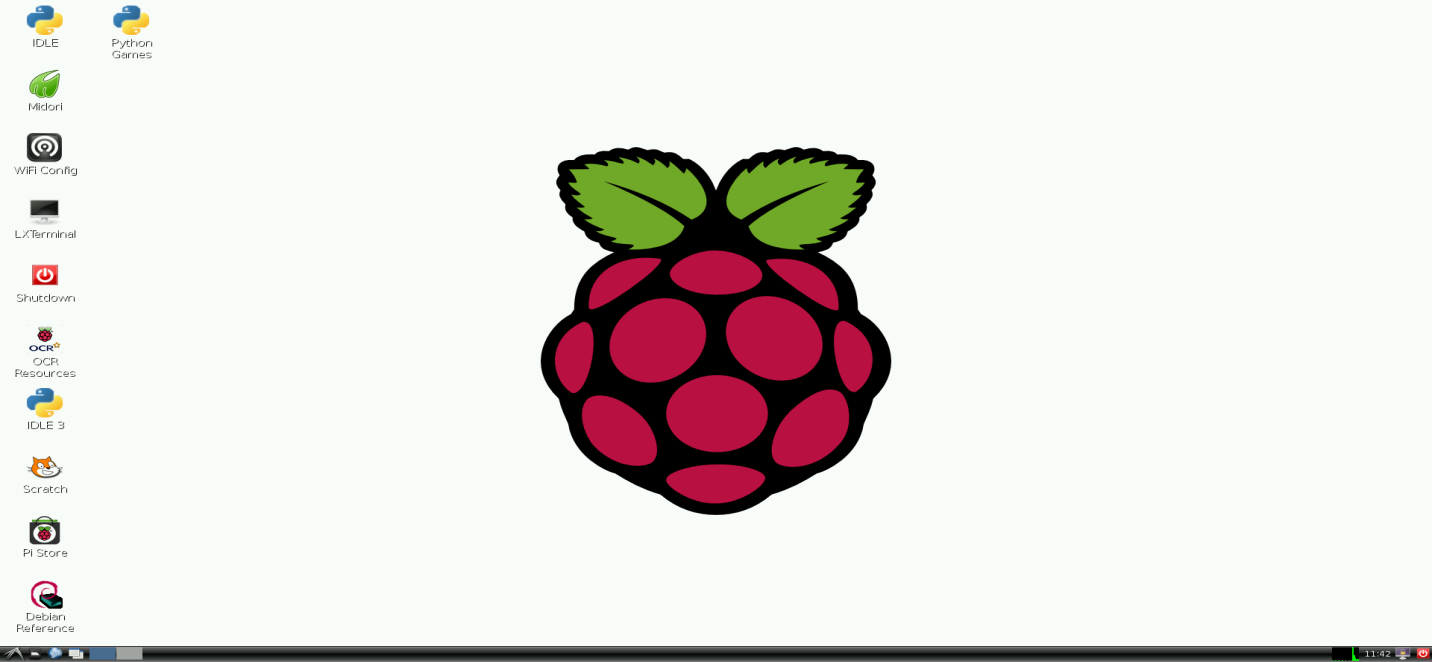
Công ty FPT Software cũng đang rất quan tâm đến lĩnh vực này và đầu tư rất nhiều vào đây. Bằng chứng là những cuộc thi về xe tự hành do FPT tổ chức được nhiều bạn sinh viên quan tâm và tham gia. Hiện nay, FPT cũng đang dần có sự hợp tác với nhiều hãng lớn trên thế giới về lĩnh vực Automotive. Tất cả đã chứng minh là trong tương lai, xe tự hành sẽ được phát triển mạnh và có thể đưa vào sử dụng chứ không chỉ còn như trong phim ảnh nữa.



**Hình 2.1:** Hình ảnh thực tế của xe tự lái Tesla

1. **Lý thuyết và các thuật toán xử lý ảnh**
   1. ***Hệ điều hành Raspbian***

Raspbian là một hệ điều hành dễ sử dụng và với sự hỗ trợ tốt của cộng đồng trên thế giới. Hơn thế nữa, Raspbian là phiên bản không chính thức của Debian Wheezy – một  phiên bản Linux có tiếng. Raspbian rất đơn giản và quen thuộc.



**Hình 2.2:** Giao diện hệ điều hành Raspbian

Giống như một chiếc máy tính Windows, Raspbian bao gồm các ứng dụng đa phương tiện và đồ họa (xem ảnh, xem phim, soạn thảo notepad), và nếu bạn cần nhiều hơn, bạn có thể cài thêm các gói phần mềm khác như trình duyệt internet, nhắn tin, bộ phần mềm văn phòng.

* 1. ***Ngôn ngữ lập trình C++***

C++ là một ngôn ngữ lập trình kiểu tĩnh, mục đích tổng quát, phân biệt kiểu chữ mà hỗ trợ lập trình hướng đối tượng, lập trình thủ tục.

C++ đuợc coi như là ngôn ngữ **bậc trung (middle-level)**, khi nó kết hợp các đặc điểm và tính năng của ngôn ngữ bậc cao và bậc thấp.

C++ đuợc phát triển bởi Bjarne Stroustrup năm 1979 tại Bell Labs ở Murray Hill, New Jersey, như là một bản nâng cao của ngôn ngữ C và với tên gọi đầu tiên là “C with classes”, nhưng sau đó đuợc đổi tên thành C++ vào năm 1983.

C++ là một Superset của C, và bất kỳ chuong trình C nào cũng là một chương trình C++.

* 1. ***Thư viện OpenCV***

OpenCV là một thư viện mã nguồn mở hàng đầu cho thị giác máy tính (computer vision), xử lý ảnh và [máy học](https://techmaster.vn/khoa-hoc/25511/machine-learning-co-ban), và các tính năng tăng tốc GPU trong hoạt động thời gian thực.



**Hình 2.3:** Logo OpenCV

OpenCV được phát hành theo giấy phép BSD, do đó nó hoàn toàn miễn phí cho cả học thuật và thương mại. Nó có các interface C++, C, Python, Java và hỗ trợ Windows, Linux, Mac OS, iOS và Android. OpenCV được thiết kế để tính toán hiệu quả và với sự tập trung nhiều vào các ứng dụng thời gian thực. Được viết bằng tối ưu hóa C/C++, thư viện có thể tận dụng lợi thế của xử lý đa lõi. Được sử dụng trên khắp thế giới, OpenCV có cộng đồng hơn 47 nghìn người dùng và số lượng download vượt quá 6 triệu lần. Phạm vi sử dụng từ nghệ thuật tương tác, cho đến lĩnh vực khai thác mỏ, bản đồ trên web hoặc công nghệ robot.

OpenCV đang được sử dụng rộng rãi trong các ứng dụng bao gồm:

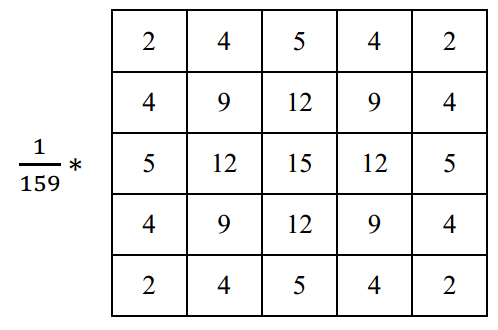
* Hình ảnh street view
* Kiểm tra và giám sát tự động
* Robot và xe hơi tự lái
* Phân tích hình ảnh y tế
* Tìm kiếm và phục hồi hình ảnh/video
* Phim - cấu trúc 3D từ chuyển động
* Nghệ thuật sắp đặt tương tác
  1. ***Thuật toán Canny***

Kỹ thuật phát hiện biên theo phương pháp Canny là phương pháp phát hiện biêntrực tiếp rất hiệu quả áp dụng cho loại ảnh nhiễu.

Phương pháp này do Canny phòng thí nghiệm MIT khởi xướng năm 1986. Canny đã đưa ra tập hợp các mục tiêu của một phương pháp phát hiện biên và đưa ra một phương pháp tối ưu đểthực hiện các mục tiêu đó. Phương pháp này gọi là phương pháp Canny

Thuật toán được tiến hành qua 4 bước cơ bản:

**B1: Giảm nhiễu**  
Thông thường để giảm nhiễu sử dụng các bộ lọc làm mờ. Có thể sử dụng bộ lọc Gaussian để tích chập với ảnh:

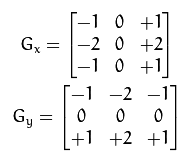


**Hình 2.4:** Bộ lọc Gauss kích thước 5x5

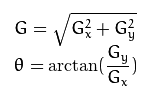
**B2: Tính độ lớn và góc của Gradient:**

Tính đạo hàm Dx ( x,y) và Dy(x,y ) theo chiều x và y của ảnh. Một số bộ lọc như: Roberts, Prewitt, Sobel.

Bộ lọc Sobel 3x3:



Độ lớn Gradient và góc Gradient:



Tính θ’ bằng cách làm tròn từ góc θ vào một trong bốn hướng: 00, 450, 900, 1350.

**B3: Chặn không cực đại (Non-Maximum Surpression)**

Bước này chỉ giữ lại những pixel thuộc cạnh mà có độ lớn gradient lớn nhất.

Xem xét 3 pixel trong vùng 3 x 3 xung quanh pixel (x,y):

* Nếu θ(x, y) = 00 thì (x+1, y), (x, y) và (x-1, y) được xem xét.
* Nếu θ(x, y) = 900 thì (x, y+1), (x, y) và (x, y-1).
* Nếu θ(x, y) = 450 thì (x+1, y+1), (x, y) và (x-1, y-1).
* Nếu θ(x, y) = 1350 thì (x-1, y+1), (x, y) và (x+1, y-1).

Nếu pixel (x, y) có gradient lớn nhất của 3 pixel xem xét thì pixel đó là cạnh.

**B4: Ngưỡng Hysteresis (Hysteresis Thresholding)**

Hysteresis sử dụng 2 ngưỡng, ngưỡng thigh và tlow. Pixel mà có độ lớn gradient D < tlow thì được loại ngay lập tức. Những pixel tlow < D < thigh được giữ lại nếu là một cạnh liên tục với những pixel có độ lớn gradient D > thigh.



**Hình 2.5:** Ảnh trước và sau khi dùng thuật toán canny

Cấu trúc hàm Canny:

Canny (detected\_edges, detected\_edges, lowThreshold, highThreshold);

Trong đó:

* + - *detected\_edges*: Ảnh gốc ở dạng xám trắng
    - *detected\_edges*: Ảnh đầu ra của bộ lọc
    - *lowThreshold*: Giá trị ngưỡng dưới để tách biên
    - *highThreshold*: Đặt trong chương trình gấp ba lần *lowThreshold* (theo khuyến cáo của Canny)
  1. ***Thuật toán HoughLines***

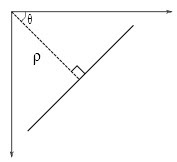
HoughLines để phát hiện đường thẳng, là thuật toán tìm đường thẳng mặc dù có thể bị đứt đoạn trong ảnh.

Trong hình học, đường thẳng có phương trình:

y = m.x + c (2.1)

hoặc là: ρ = x.cosθ + y.sinθ (2.2)

Với ρ là khoảng cách từ đường thẳng tới gốc tọa độ, còn θ là góc giữa trục hoành và đoạn thẳng ngắn nhất nối tới gốc tọa độ (đơn vị là radian).

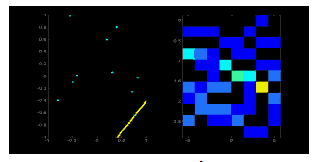


**Hình 2.6:** Hình minh họa trong OpenCV

Có 2 cách để xác định 1 đường thẳng, người ta chọn cách thứ 2 vì chỉ cần 2 tham số (ρ, θ) tiện cho tính toán hơn. Vì θ được tính bằng radian nên có giá trị trong khoảng [0;π] (hay là [0;3.14]). Khi θ bằng 0 hoặc bằng 3.14 thì đường thẳng dựng đứng, còn θ bằng 1.57 (π/2) là nằm ngang.

**Mô tả cách hoạt động của thuật toán**

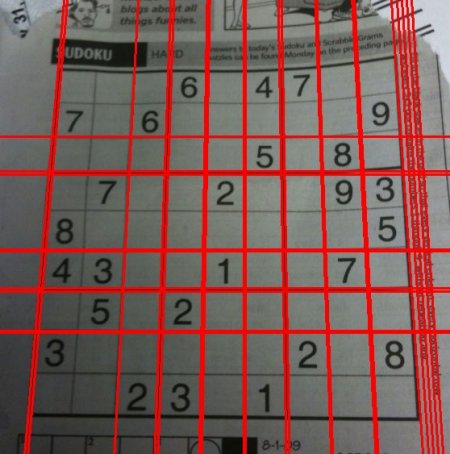
Vì mỗi đường thẳng được xác dịnh bởi 2 giá trị (ρ, θ) nên thuật toán tạo 1 mảng 2 chiều). Dòng ứng với ρ và cột ứng với θ, kích thước của mảng có thể tùy chọn chọn, là mảng càng lớn thì càng chính xác và tính toán càng lâu, còn mảng nhỏ thì nhanh hơn nhưng không chính xác bằng.



**Hình 2.7:** Hình ảnh mô tả hoạt động của thuật toán hough line

Trong Ảnh: bên trái là các điểm trong ảnh, bên phải là mảng 2 chiều

Khi sử dụng người dùng sẽ truyền giá trị ρ và θ họ mong muốn(Thường thì người ta sẽ truyền rho = 1px, theta = 3.14/180 (rad)). Thuật toán sẽ vẽ ρ x θ đường thẳng trên ảnh, với mỗi đường thẳng thuật toán đếm số pixel nằm trên đường thẳng đó, cứ mỗi pixel tìm thấy thì cộng thêm giá trị vào ô ứng với (ρ, θ).



**Hình 2.8:** Kết quả thu được sau khi dùng thuật toán HoughLine

Cú pháp hàm Houghline:

HoughLines (InputArray image, OutputArray lines, double rho, double theta, int threshold, double srn=0, double stn=0)

* image – ảnh input nhị phân (ảnh có thể bị thay đổi khi tính toán)
* lines – Output là vector chứa các đường thẳng. Mỗi đường thẳng chứa 2 giá trị (ρ, θ)
* rho – Khoảng cách của các đường thẳng tính bằng pixels
* theta – Khoảng cách góc của các đường thẳng tính bằng radians
* threshold – Đường thẳng nào có số pixel lớn hơn threshold mới lấy
* srn – Dùng cho multi-scale
* stn – Cũng dùng cho multi-scale

Thường thì người ta sẽ truyền rho = 1px, theta = 3.14/180 (rad)

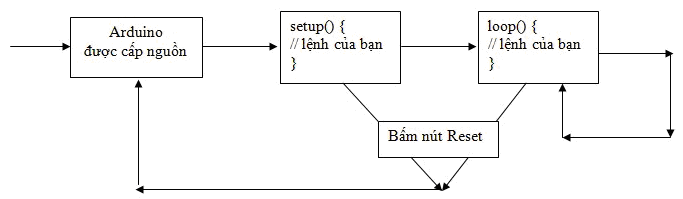
1. **Lý thuyết về điều khiển motor và servo**
2. ***Ngôn ngữ lập trình***

Như đã nói từ chương 1, việc điều khiển motor và servo sẽ thông qua Arduino UNO R3, do đó ngôn ngữ được sử dụng là ngôn ngữ dành riêng cho Arduino trên Arduino IDE được gọi là ngôn ngữ Arduino. Ngôn ngữ Arduino được phát triển dựa trên nền tảng của C, tuy nhiên nó có cấu trúc hoàn toàn khác và rất dễ sử dụng. Một chương trình Arduino gồm 2 phần chính là setup() và loop(). Những lệnh trong setup() sẽ được chạy khi chương trình của bạn khởi động. Bạn có thể sử dụng nó để khai báo giá trị của biến, khai báo thư viện, thiết lập các thông số,…

Sau khi setup() chạy xong, những lệnh trong loop() được chạy. Chúng sẽ lặp đi lặp lại liên tục cho tới khi nào bạn ngắt nguồn của board Arduino mới thôi.

Bất cứ khi nào bạn nhất nút Reset, chương trình của bạn sẽ trở về lại trạng thái như khi Arduino mới được cấp nguồn[.](http://arduino.vn/reference/setup-va-loop)

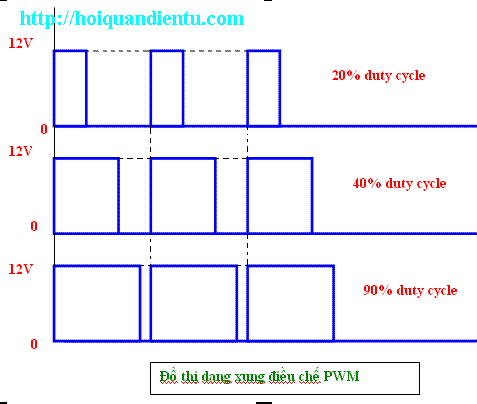
Quá trình này có thể được miêu tả như sơ đồ dưới đây



**Hình 2.9:** Sơ đồ hoạt động của Arduino

***2.4.2 Phương pháp điều xung PWM***

Phương pháp điều xung PWM (Pulse Width Modulation) là phương pháp điều chỉnh điện áp ra tải, hay nói cách khác, là phương pháp điều chế dựa trên sự thay đổi độ rộng của chuỗi xung vuông, dẫn đến sự thay đổi điện áp ra.  
Các PWM khi biến đổi thì có cùng 1 tần số và khác nhau về độ rộng của sườn dương hay sườn âm.



**Hình 2.10:** Đồ thị dạng xung điều chế PWM

**Ứng dụng của PWM trong điều khiển**

PWM được ứng dụng nhiều trong điều khiển. Điển hình nhất mà chúng ta thường hay gặp là điều khiển động cơ và các bộ xung áp, điều áp... Sử dụng PWM điều khiển độ nhanh chậm của động cơ hay cao hơn nữa, nó còn được dùng để điều khiển sự ổn định tốc độ động cơ.

Ngoài lĩnh vực điều khiển hay ổn định tải thì PWM còn tham gia và điều chế các mạch nguồn như : boot, buck, nghịch lưu 1 pha và 3 pha...

PWM còn gặp nhiều trong thực tế ở các mạch điện điều khiển. Điều đặc biệt là PWM chuyên dùng để điều khiển các phần tử điện tử công suất có đường đặc tính là tuyến tính khi có sẵn 1 nguồn 1 chiều cố định. Như vậy PWM được ứng dụng rất nhiều trong các thiết bị điện- điện tử. PWM cũng chính là nhân tố mà các đội Robocon sử dụng để điều khiển động cơ hay ổn định tốc độ động cơ.

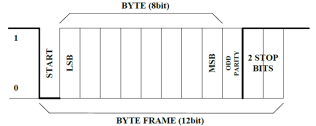
## 2.5 Giao tiếp giữa Raspberry và Arduino

Như đã nói ở trên, hệ thống của chúng tôi gồm hai phần chính được điều khiển bằng Raspberry và Arduino. Do đó, cần có một giao tiếp để kết nối giữa hai phần này và mặc dù có rất nhiều cách để kết nối nhưng ở đây chúng tôi sử dụng giao tiếp UART cho việc liên lạc giữa Raspberry và Arduino.

**Thế nào là giao tiếp UART**

UART là viết tắt của Universal Asynchronous Receiver – Transmitter. Thường là một mạch tích hợp được sử dụng trong việc truyền dẫn dữ liệu nối tiếp giữa máy tính và các thiết bị ngoại vi. Rất nhiều vi điều khiển hiện nay đã được tích hợp UART, vì vấn đề tốc độ và độ điện dụng của UART không thể so sánh với các giao tiếp mới hiện nay nên các dòng PC & Laptop đời mới không còn tích hợp cổng UART.

Như đã biết giao tiếp SPI và I2C có 1 dây truyền dữ liệu và 1 dây được sử dụng để truyền xung clock (SCL) để đồng bộ trong giao tiếp. Với UART thì không có dây SCL, vấn đề được giải quyết khi mà việc truyền UART được dùng giữa 2 vi xử lý với nhau, đồng nghĩa với việc mỗi vi xử lý có thể tự tạo ra xung clock cho chính nó xử dụng. Để bắt đầu cho việc truyền dữ liệu bằng UART, một **START bit** được gửi đi, sau đó là các bit dữ liệu và kết thúc quá trình truyền là **STOP bit**.



**Hình 2.11:** Sơ đồ truyền dữ liệu bằng UART

Như hình có thể thấy, khi ở trạng thái chờ mức điện thế ở mức 1 (high). Khi bắt đầu truyền **START bit** sẻ chuyển từ 1 xuống 0 để báo hiệu cho bộ nhận là quá trình truyền dữ liệu sắp xảy ra. Sau **START bit** là đến các bit dữ liệu D0-D7 (Theo hình vẽ các bit này có thể ở mức High or Low tùy theo dữ liệu). Sau khi truyền hết dữ liệu thì đến **Bit Parity** để bộ nhận kiểm tra tính đúng đắn của dữ liệu truyền (vấn đề này mình sẽ giải thích rõ hơn trong tài liệu CRC trong thời gian tới). Cuối cùng là **STOP bit** là 1 báo cho thiết bị rằng các bit đã được gửi xong. Thiết bị nhận sẽ tiến hành kiểm tra khung truyền nhằm đảm báo tính đúng đắn của dữ liệu.  
**Các thông số cơ bản trong truyền nhận UART:**

- **Baund rate (tốc độ baund ):** Khoảng thời gian dành cho 1 bit được truyền. Phải được cài đặt giống nhau ở gửi và nhận.

**- Frame (khung truyền ):** Khung truyền quy định về số bit trong mỗi lần truyền.

- **Start bit:** là bit đầu tiên được truyền trong 1 Frame. Báo hiệu cho thiết bị nhận có một gói dữ liệu sắp đc truyền đến. Bit bắt buộc.

- **Data:** dữ liệu cần truyền. Bit có trọng số nhỏ nhất LSB được truyền trước sau đó đến bit MSB.

- **Parity bit:** kiểm tra dữ liệu truyền có đúng không.

- **Stop bit:** là 1 hoặc các bit báo cho thiết bị rằng các bit đã được gửi xong. Thiết bị nhận sẽ tiến hành kiểm tra khung truyền nhằm đảm bảo tính đúng đắn của dữ liệu. bit bắt buộc.

**2.6 Kết luận chương**

Như vậy, ở chương này, chúng ta đã điểm qua những cơ sở lý thuyết mà sẽ được sử dụng. Từ những cơ sở này, ta sẽ xây dựng những thuật toán để giải quyết những bài toán đặt ra mà ta sẽ trình bày và làm rõ ở chương sau.

# Chương 3: THIẾT KẾ CÁC THUẬT TOÁN

**Giới thiệu chương**

**Sơ đồ hệ thống**

Khối điều khiển động cơ (Module L298)

Khối thu nhận hình ảnh (Camera)

Khối xử lý hình ảnh (Raspberry PI 3)

Động cơ (DC Motor )

Khối hiển thị (LCD-Touchscreen)

Khối điều khiển hướng di chuyển (Module Servo)

Nguồn A

Nguồn B

Khối điều khiển xe (Arduino UNO R3)

**Hình 2.1:** Sơ đồ khối hệ thống

**Thiết kế và kết nối các thành phần của hệ hống**

***Raspberry PI 3 và Camera***

***Raspberry PI 3 và Arduino UNO R3***

***Giao tiếp Serial***

***Sơ đồ kết nối***

***Arduino UNO R3 và Servo***

***Lý thuyết điều chế xung PWM***

***Sơ đồ kết nối***

***Arduino UNO R3 và Bộ điều tốc ESC***

***Arduino UNO R3 và Cảm biến từ trường HALL***

1. **Lý thuyết và các thuật toán xử lý ảnh**
   * 1. ***Thư viện OpenCV***
     2. ***Thuật toán Canny***
     3. ***Thuật toán HoughLines***
     4. ***Thuật toán xác định làn đường***
     5. ***Thuật toán xác định giao điểm đường thẳng và tính góc***
   1. **Lý thuyết và các thuật toán điều khiển xe**
      1. ***Thuật toán điều khiển hướng di chuyển***
      2. ***Thuật toán điều khiển tốc độ động cơ***
      3. ***Thuật toán xác định vận tốc của xe***
   2. **Sơ đồ thuật toán**
      1. ***Sơ đồ thuật toán chung***
      2. ***Sơ đồ thuật toán từng khối***
   3. **Kết luận chương**

# Chương 4: VẬN HÀNH HỆ THỐNG

1. **Giới thiệu chương**
2. **Mô hình hệ thống**
3. **Vận hành**
   * 1. ***Kiểm tra nhận dạng làn đường và xác định góc lệch***
     2. ***Kiểm tra phần điểu khiển xe***

***Kiểm tra phần điều khiển tốc dộ động cơ***

***Kiểm tra phần điều khiển hướng di chuyển***

***Kiểm tra phần đo tốc độ xe***

* + 1. ***Kiểm tra vận hành toàn bộ hệ thống***

1. **Kết quả đạt được**
2. **Kết luận chương**

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

# TÀI LIỆU THAM KHẢO

# PHỤ LỤC