TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

PROJECT

**Kỹ thuật ghép nối  
máy tính**

**Ô TÔ QUAN TRẮC MÔI TRƯỜNG   
ĐIỀU KHIỂN QUA WIFI  
TRÊN SERVER DI ĐỘNG**

Sinh viên thực hiện: **Đào Đức Tùng  
 Lê Xuân An**

Lớp KTMT – K57

Giáo viên hướng dẫn: Th.S. **Bùi Quốc Anh**

HÀ NỘI 01 -2016

# THÔNG TIN PROJECT

**1. Thông tin về sinh viên**

Họ và tên sinh viên: Đào Đức Tùng – 20122731

Lê Xuân An - 20124854

Lớp: Kỹ thuật máy tính K57 Hệ đào tạo: Đại học chính quy

**2. Sản phẩm mục tiêu**

Ô tô quan trắc môi trường điều khiển qua wifi trên server di động

**3. Các tính năng**

* Sử dụng Github để lưu trữ các phiên bản của project

<https://github.com/tungdao17/Wifi-motor/>

**Các tính năng tùy chọn:**

|  |
| --- |
| Tính năng |
| Sử dụng hệ thống bánh lái bằng servo |
| Có Mode sử dụng hệ thống bánh lái bằng động cơ sau |
| Hiển thị dữ liệu lên LCD trên xe |
| Điều khiển xe bằng phần mềm trên máy tính qua wifi |
| Trao đổi dữ liệu với máy tính qua wifi hoặc UART |
| Xe có thể tự tạo server di động theo xe |
| Xe có thể kết nối đến wifi khác và vẫn có server lắng nghe trong mạng đó |
| Sử dụng UART để truyền nhận dữ liệu giữa Module Wifi với VĐK |
| Có hệ thống nút nguồn, reset |
| Có thể gắn thêm hệ thống các cảm biến quan trắc môi trường |
| Có thể giao tiếp với các hệ thống khác thông qua một số giao thức |

**MỤC LỤC**

[THÔNG TIN PROJECT 2](#_Toc440527223)

[MỤC LỤC 3](#_Toc440527224)

[DANH MỤC CÁC TỪ VIẾT TẮT VÀ THUẬT NGỮ 4](#_Toc440527225)

[Tài liệu tham khảo 4](#_Toc440527226)

[PHẦN I: NỀN TẢNG VÀ CÔNG NGHỆ PHÁT TRIỂN ỨNG DỤNG 5](#_Toc440527227)

[1.1. Công nghệ kỹ thuật hỗ trợ liên quan 5](#_Toc440527228)

[1.1.1. Microsoft Visual Studio 5](#_Toc440527229)

[1.1.2. GitHub 5](#_Toc440527230)

[1.1.3. CodevisionAVR 5](#_Toc440527231)

[1.1.4. Progisp 6](#_Toc440527232)

[1.1.5. Nodemcu Firmware Programmer 6](#_Toc440527233)

[1.1.6. ESPlorer 6](#_Toc440527234)

[1.1.7. Altium Designer 7](#_Toc440527235)

[1.1.8. SolidWorks 8](#_Toc440527236)

[1.2. ATmega16 8](#_Toc440527237)

[1.3. Module ESP8266 11](#_Toc440527238)

[1.4. Text LCD 13](#_Toc440527239)

[1.5. Module L298 14](#_Toc440527240)

[1.6. Mini Servo ES08A 15](#_Toc440527241)

[1.7. Module USB to COM CP2102 16](#_Toc440527242)

[1.8. IC LM2576 16](#_Toc440527243)

[PHẦN 2: PHÂN TÍCH THIẾT KẾ 18](#_Toc440527244)

[2.1. Mô hình ghép nối và liên kết 18](#_Toc440527245)

[2.1.1. Mô hình ghép nối phần cứng 18](#_Toc440527246)

[2.1.2. Mô hình phần mềm điều khiển 19](#_Toc440527247)

[2.1.3. Mạch nguồn 19](#_Toc440527248)

[2.1.4. Thiết kế cơ khí xe 20](#_Toc440527249)

[2.2. Thông số kỹ thuật 22](#_Toc440527250)

[2.3. Các hàm chức năng 23](#_Toc440527251)

[2.3.1. Atmega16 23](#_Toc440527252)

[2.3.2. Module ESP8266 24](#_Toc440527253)

[2.3.3. Phần mềm điều khiển 25](#_Toc440527254)

[PHẦN 3: SẢN PHẨM KẾT QUẢ 26](#_Toc440527255)

[3.1. Giao diện phần mềm điều khiển 26](#_Toc440527256)

[3.2. Sản phẩm Xe 27](#_Toc440527257)

**DANH MỤC CÁC TỪ VIẾT TẮT VÀ THUẬT NGỮ**

|  |  |  |
| --- | --- | --- |
| **Số thứ tự** | **Từ viết tắt** | **Ý nghĩa** |
| 1 | MS | Microsoft |
| 2 | VS | Visual Studio |
| 3 | UART | Universal Asynchronous serial Reveiver and Transmitter |
| 4 | T/C | Timer/Counter |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |

**Tài liệu tham khảo**

1. Giáo trình Kỹ thuật ghép nối máy tính – ThS. Bùi Quốc Anh, Bộ môn Kỹ thuật máy tính, Viện Công nghệ thông tin và Truyền thông, trường Đại học Bách khoa Hà Nội
2. Bài giảng Lập trình mạng – ThS. Lương Ánh Hoàng, Bộ môn Kỹ thuật máy tính, Viện Công nghệ thông tin và Truyền thông, trường Đại học Bách khoa Hà Nội
3. Datasheet Atmega16, ESP8266, CP2102, L298, LM2576
4. Một số trang web:

* http://hocavr.com/
* http://www.esp8266.com/
* <https://github.com/esp8266/esp8266-wiki/wiki>/
* <http://lua-users.org/wiki/>
* https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu\_api\_en/
* https://dientunguyentuan.wordpress.com/2015/05/15/bai-4-lap-trinh-dieu-khien-esp8266/

# PHẦN I: NỀN TẢNG VÀ CÔNG NGHỆ PHÁT TRIỂN ỨNG DỤNG

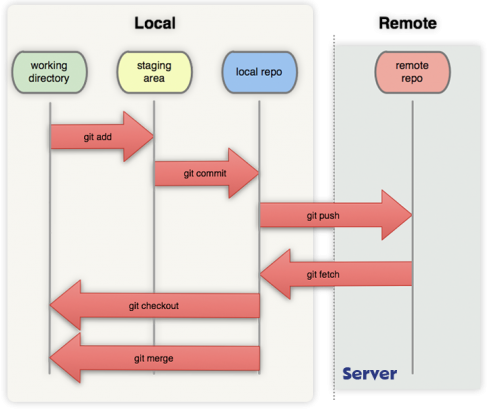
## Công nghệ kỹ thuật hỗ trợ liên quan

### Microsoft Visual Studio

Microsoft Visual Studio là một môi trường phát triển tích hợp (IDE) từ Microsoft. Nó được sử dụng để phát triển chương trình máy tính cho Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Visual Studio sử dụng nền tảng phát triển phần mềm của Microsoft như Windows API, Windows Forms, Windows Presentation Foundation, Windows Store và Microsoft Silverlight. Nó có thể sản xuất cả hai ngôn ngữ máy và mã số quản lý.

Trong project này, MS Visual Studio được dùng để lập trình phần mềm điều khiển xe với ngôn ngữ C#.

### GitHub

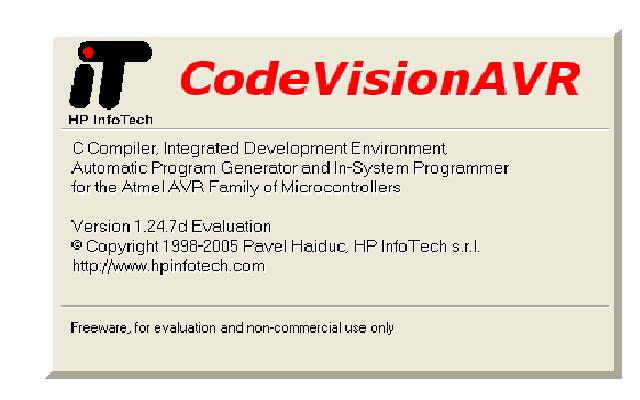
Github [http://github.com](http://github.com/), còn được gọi là social network dành cho developer đi vào hoạt động tháng 2 năm 2008, là một dịch vụ sử dụng hệ thống quản lý phân tán GIT giúp người dùng lưu trữ source code cho các dự án.

GitHub cung cấp dịch vụ thương mại và cả tài khoản miễn phí cho các dự án nguồn mở. Theo khảo sát của người sử dụng Git vào năm 2009, Github hiện đang là server Git lưu trữ source code phổ biến nhất hiện nay

Trong project này, GitHub được dùng để lưu trữ mã nguồn và báo cáo môn học.

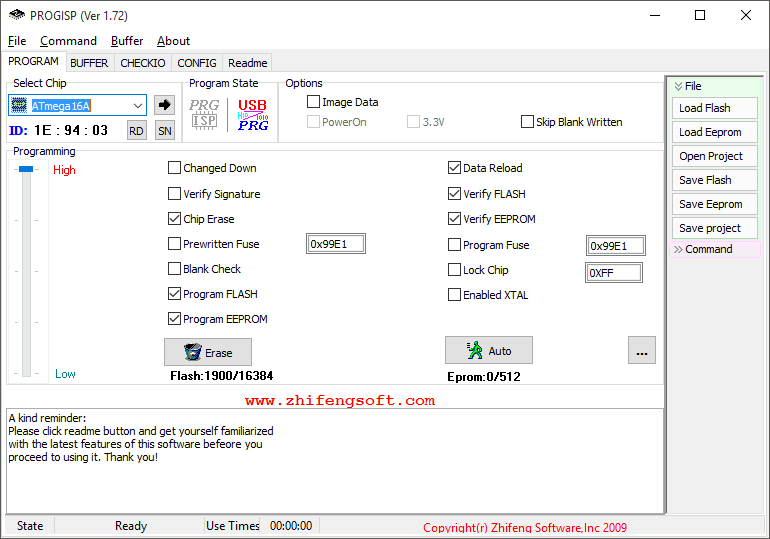
### CodevisionAVR

CodeVisionAVR - là một môi trường phát triển tích hợp phần mềm cho vi điều khiển Atmel AVR. Nó cung cấp sự hỗ trợ rộng rãi cho các thiết bị AVR và tạo ra một đoạn mã nhỏ gọn và hiệu quả.

CodeVisionAVR - một IDE với trình biên dịch C, trình hợp dịch cho dòng chip của AVR. Với 2 phiên bản free cung cấp cho chúng ta một môi trường lập trình khá tốt cho dòng vi xử lý của AVR

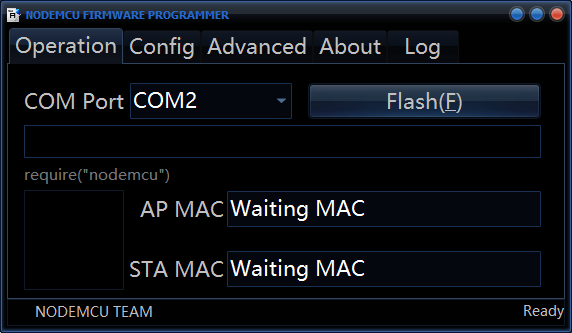
Trong project này, CodeVision được dùng để lập trình cho VĐK với ngôn ngữ C.

### Progisp

Phần mềm do zhifengsoft.com phát triển, chuyên để nạp code, fuse bit cho VĐK, đặc biệt là dòng chip do Atmel sản xuất

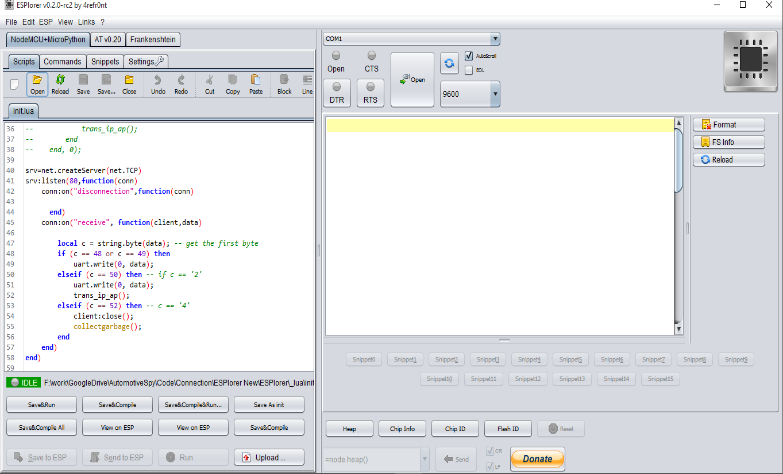
Trong project này, Progisp được dùng để nạp code cho ATmega16

### Nodemcu Firmware Programmer

Nodemcu là một công cụ mạnh mẽ cho các kỹ sư phần mềm chương trình từ HardwareIO - [www.nodemcu.com](http://www.nodemcu.com)

Phần mềm này chuyên dùng để nạp Firmware cho Module ESP8266

### ESPlorer

ESPlorer là phần mềm do 4refr0nt - <http://esp8266.ru/esplorer/> phát triển

Phần mềm này chuyên dùng để nạp code và lập trình cho Module ESP8266

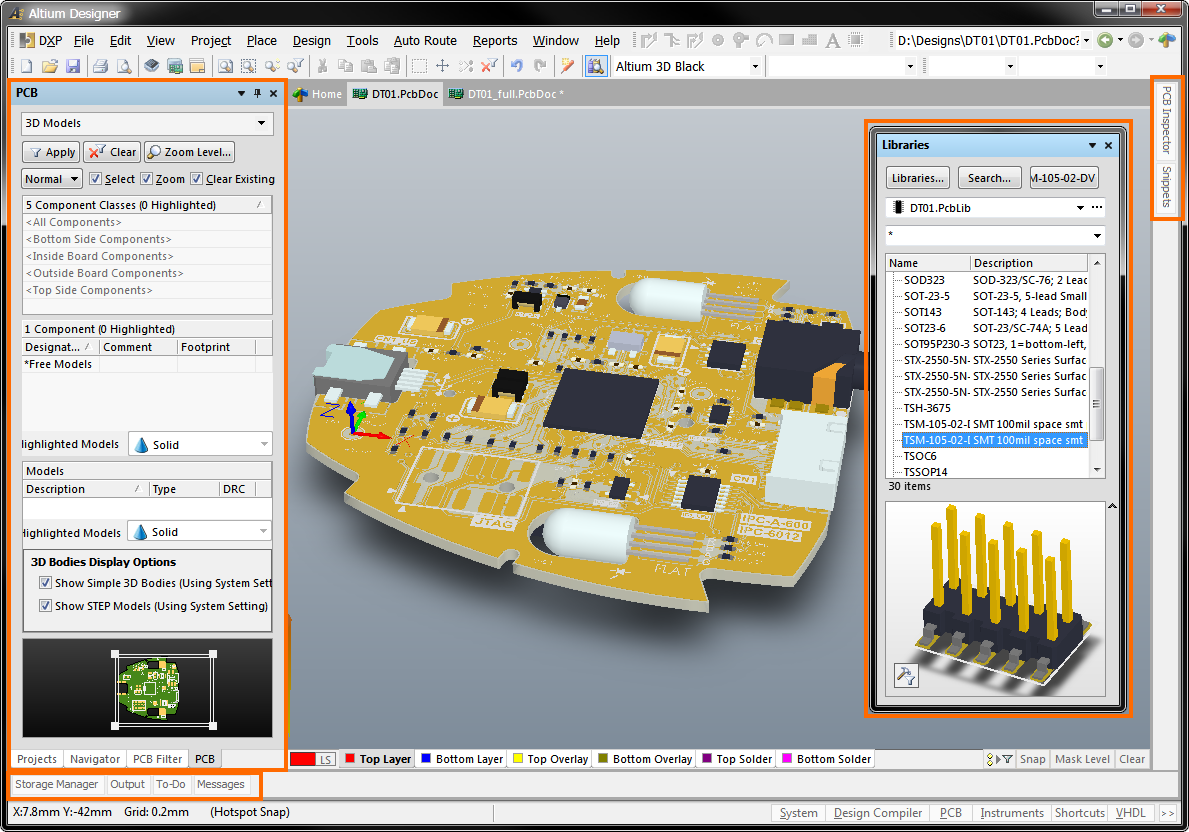
Trong project này sử dụng ESPlorer để lập trình cho Module ESP8266 bằng ngôn ngữ Lua.

### Altium Designer

Phần mềm thiết kế mạch tự động Altium Designer là một môi trường thiết kế điện tử đồng nhất, tích hợp cả thiết kế nguyên lý, thiết kế mạch in PCB, lập trình hệ thống nhúng và FPGA.

Các điểm đặc trưng của Altium Designer :

* Giao diện thiết kế, quản lý và chỉnh sửa thân thiện, dễ dàng biên dịch, quản lý file, quản lý phiên bản cho các tài liệu thiết kế.
* Hỗ trợ mạnh mẽ cho việc thiết kế tự động, đi dây tự động theo thuật toán tối ưu, phân tích lắp ráp linh kiện. Hỗ trợ việc tìm các giải pháp thiết kế hoặc chỉnh sửa mạch, linh kiện, netlist có sẵn từ trước theo các tham số mới.
* Mở, xem và in các file thiết kế mạch dễ dàng với đầy đủ các thông tin linh kiện, netlist, dữ liệu bản vẽ, kích thước, số lượng…
* Hệ thống các thư viện linh kiện phong phú, chi tiết và hoàn chỉnh bao gồm tất cả các linh kiện nhúng, số, tương tự…
* Đặt và sửa đối tượng trên các lớp cơ khí, định nghĩa các luật thiết kế, tùy chỉnh các lớp mạch in, chuyển từ schematic sang PCB, đặt vị trí linh kiện trên PCB.
* Mô phỏng mạch PCB 3D, đem lại hình ảnh mạch điện trung thực trong không gian 3 chiều, hỗ trợ MCAD-ECAD, liên kết trực tiếp với mô hình STEP, kiểm tra khoảng cách cách điện, cấu hình cho cả 2D và 3D
* Hỗ trợ thiết kế PCB sang FPGA và ngược lại.



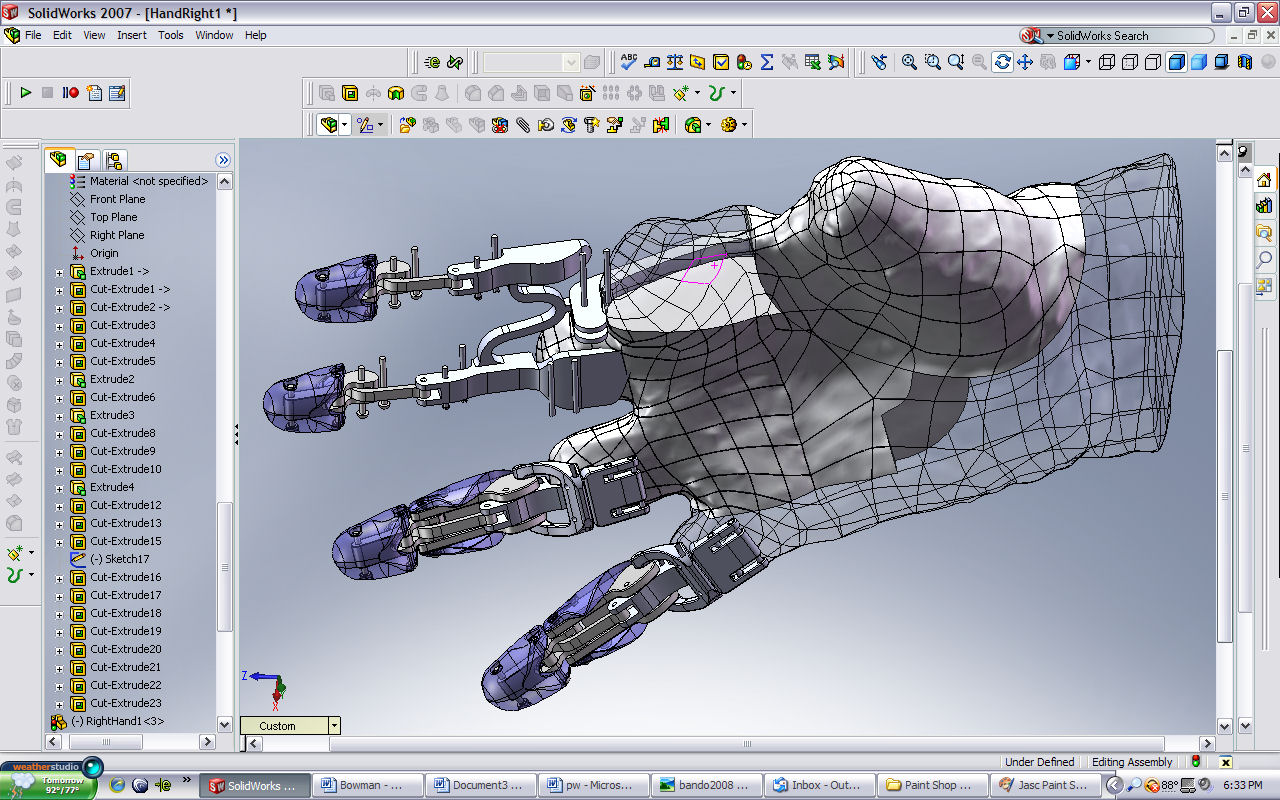
Trong project này, Altium Designer được sử dụng để thiết kế và vẽ mạch in

### SolidWorks

Solidworks là phần mềm thiết kế cơ khí 3D, chạy trên hệ điều hành Windowns. Solidworks được phát triển bởi Dassault Systèmes Solidworks Corp, một công ty con của Dassault Systèmes, SA (Vélizy, Pháp).

Solidworks xây dựng mô hình dựa trên cách tiếp cận thành phần-tham số để tạo mô hình và lắp ráp. Các tham số có giá trị xác định hình dạng hoặc hình học của mô hình và lắp ráp. Thông số có thể là các thông số số, chẳng hạn như độ dài đoạn thẳng hoặc đường kính vòng tròn, hoặc các thông số hình học, chẳng hạn như tiếp tuyến, song song, đồng tâm, ngang hay dọc, vv… Thông số số có thể được liên kết với nhau thông qua việc sử dụng các mối quan hệ, đáp ứng theo ý định thiết kế.

SolidWorks cũng bao gồm các tính năng bổ sung được tích hợp tiên tiến như thiết bị và bộ phận cam, cho phép lắp ráp thiết bị mô hình để tạo chính xác chuyển động quay của bánh răng như thực tế.



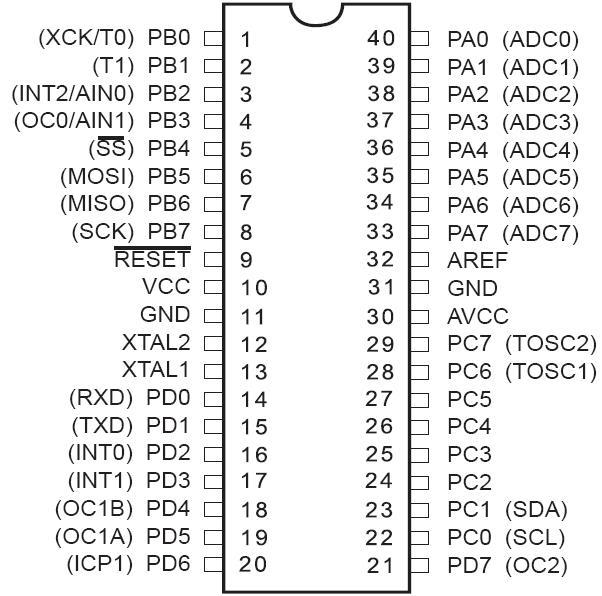
Trong project này, SolidWorks được sử dụng để thiết kế sơ bộ bộ khung của xe, cơ cấu bánh lái và các chi tiết của bộ phận lái

## ATmega16

ATmega16 là bộ vi điều khiển CMOS 8 bit tiêu thụ điện năng thấp dựa trên kiến trúc RISC Reduced Instruction Set Computer.

Với công nghệ này cho phép các lệnh thực thi chỉ trong một chu kì xung nhịp, vì thế tốc độ xử lý dữ liệu có thể đạt đến 1 triệu lệnh trên giây ở tần số 1 Mhz.

Atmega16 có tập lệnh phong phú về số lượng với 32 thanh ghi làm việc đa năng. Toàn bộ 32 thanh ghi đều được nối trực tiếp với ALU (Arithmetic Logic Unit), cho phép truy cập 2 thanh ghi độc lập bằng một chu kì xung nhịp. Kiến trúc đạt được có tốc độ xử lý nhanh gấp 10 lần vi điều khiển dạng CISC (Complex Intruction Set Computer) thông thường.

Atmega16 gồm có 40 chân:

* Chân 1 đến 8 : Cổng nhập xuất dữ liệu song song B ( PORTB ) nó có thể đc sử dụng các chức năng đặc biệt thay vì nhập xuất dữ liệu
* Chân 9 : RESET để đưa chip về trạng thái ban đầu
* Chân 10 : VCC cấp nguồn nuôi cho vi điều khiển
* Chân 11,31 : GND 2 chân này đc nối với nhau và nối đất
* Chân 12,13 : 2 chân XTAL2 và XTAL1 dùng để đưa xung nhịp từ bên
* ngoài vào chip
* Chân 14 đến 21 : Cổng nhập xuất dữ liệu song song
* D ( PORTD ) nó có thể đc sử dụng các chức năng đặc biệt thay vì nhập xuất dữ liệu
* Chân 22 đến 29 : Cổng nhập xuất dữ liệu song song C ( PORTC ) nó có thể đc sử dụng các chức năng đặc biệt thay vì nhập xuất dữ liệu
* Chân 30 : AVCC cấp điện áp so sánh cho bộ-
* Chân 33 đến 40 : Cổng vào ra dữ liệu song song A ( PORTA ) ngoài ra nó còn đc tích hợp bộ chuyển đổi tín hiệu tương tự sang tín hiệu số ADC ( analog to digital converte)

Trong project này, Atmega16 được sử dụng làm VĐK chính, xử lý dữ liệu nhận được từ máy tính do module wifi nhận được truyền sang, điều khiển xe:

* Sử dụng UART để giao tiếp

Thuật ngữ USART trong tiếng anh là viết tắt của cụm từ: Universal Synchronous & Asynchronous serial Reveiver and Transmitter, nghĩa là bộ truyền nhận nối tiếp đồng bộ và không đồng bộ. Khái niệm USART (hay UART nếu chỉ nói đến bộ truyền nhận không đồng bộ) thường để chỉ thiết bị phần cứng (device, hardware), không phải chỉ một chuẩn giao tiếp. USART hay UART cần phải kết hợp với một thiết bị chuyển đổi mức điện áp để tạo ra một chuẩn giao tiếp nào đó.

Để quá trình truyền thành công thì việc tuân thủ các tiêu chuẩn truyền là hết sức quan trọng:

* Baud rate (tốc độ Baud): để việc truyền và nhận không đồng bộ xảy ra thành công thì các thiết bị tham gia phải “thống nhất” nhau về khoảng thời dành cho 1 bit truyền, tốc độ này gọi là tốc độ Baud – là số bit truyền trong 1 giây.
* Frame (khung truyền): là một yếu tốc quan trọng tạo nên sự thành công khi truyền và nhận, bao gồm các quy định về số bit trong mỗi lần truyền, các bit “báo” như bit Start và bit Stop, các bit kiểm tra như Parity, ngoài ra số lượng các bit trong một data cũng được quy định bởi khung truyền.
* Start bit: bit đầu tiên được truyền trong một frame truyền, bit này có chức năng báo cho thiết bị nhận biết rằng có một gói dữ liệu sắp được truyền tới. Ở module USART trong AVR, đường truyền luôn ở trạng thái cao khi nghỉ (Idle), nếu một chip AVR muốn thực hiện việc truyền dữ liệu nó sẽ gởi một bit start bằng cách “kéo” đường truyền xuống mức 0. Như vậy, với AVR bit start là mang giá trị 0 và có giá trị điện áp 0V (với chuẩn RS232 giá trị điện áp của bit start là ngược lại). start là bit bắt buộc phải có trong khung truyền.
* Data: dữ liệu cần truyền là thông tin chính cần gởi và nhận. Data là gói 5, 6, 7, 8 hoặc 9 (tương tự cho hầu hết các thiết bị hỗ trợ UART khác). Trong truyền thông nối tiếp UART, bit có ảnh hưởng nhỏ nhất (LSB – Least Significant Bit, bit bên phải) của data sẽ được truyền trước và cuối cùng là bit có ảnh hưởng lớn nhất (MSB – Most Significant Bit, bit bên trái).
* Parity bit: bit dùng kiểm tra dữ liệu truyền đúng không (một cách tương đối). Có 2 loại parity là parity chẵn (even parity) và parity lẻ (odd parity).
* Stop bits: stop bits là một hoặc các bit báo cho thiết bị nhận rằng một gói dữ liệu đã được gởi xong. Sau khi nhận được stop bits, thiết bị nhận sẽ tiến hành kiểm tra khung truyền để đảm bảo tính chính xác của dữ liệu. Stop bits là các bits bắt buộc xuất hiện trong khung truyền, trong AVR USART có thể là 1 hoặc 2 bits (Trong các thiết bị khác Stop bits có thể là 2.5 bits).

Khung truyền phổ biến nhất là : start bit+ 8 bit data+1 stop bit

Phương cách truyền nhận dữ liệu được mô tả trong datasheet.

* Sử dụng Timer để điều xung PWM

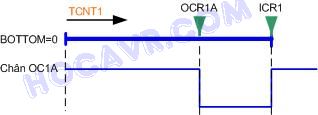
T/C là các module độc lập với CPU. Chức năng chính của các bộ T/C, như tên gọi của chúng, là định thì (tạo ra một khoảng thời gian, đếm thời gian…) và đếm sự kiện. Trên các chip AVR, các bộ T/C còn có thêm chức năng tạo ra các xung điều rộng PWM (Pulse Width Modulation), ở một số dòng AVR, một số T/C còn được dùng như các bộ canh chỉnh thời gian (calibration) trong các ứng dụng thời gian thực.

Trong project này T/C1 được sử dụng để tạo xung PWM

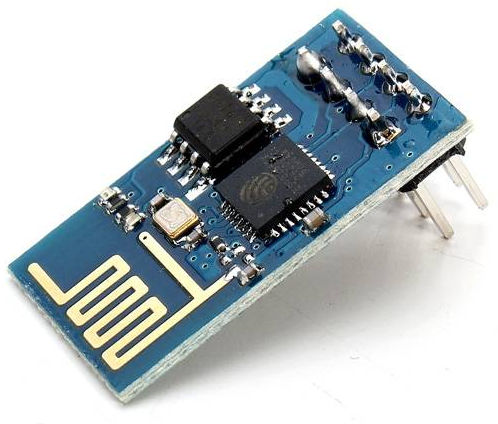
T/C1: là bộ định thời, đếm đa năng 16 bit. Bộ T/C này có 5 chế độ hoạt động chính. Ngoài các chức năng thông thường, T/C1 còn được dùng để tạo ra xung điều rộng PWM dùng cho các mục đích điều khiển. Có thể tạo 2 tín hiệu PWM độc lập trên các chân OC1A (chân 15) và OC1B (chân 16) bằng T/C1.

T/C1 có nhiều Mode, trong project này sử dụng mode Fast PWM.

2 thanh ghi TCCR1A và TCCR1B được dùng để set Mode cho T/C và set Mode cho các kênh PWM (có 2 kênh tương ứng pin ra là OC1A và OC1B)

Trong mode 14 – Fast PWM, giá trị TOP (cũng là chu kỳ của PWM) được chứa trong thanh ghi ICR1, khi hoạt động thanh ghi TCNT1 (chứa giá trị vận hành của T/C1) tăng giá trị từ 0, giả sử các bit phụ COM1A=1, COM1A0=0 (là mode “Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at TOP”), lúc này trạng thái của chân OC1A (chân 15) là HIGH (5V), khi TCNT1 tăng đến bằng giá trị của thanh ghi OCR1A thì chân OC1A được xóa về mức LOW (0V), thanh ghi đếm TCNT1 vẫn tiếp tục tăng đến khi nào nó bằng giá trị TOP chứa trong thanh ghi ICR1 thì TCNT1 tự động reset về 0 và chân OC1A trở về trạng thái HIGH

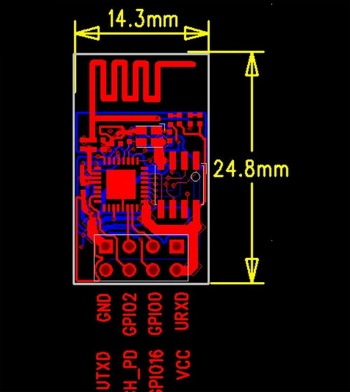
## Module ESP8266

ESP8266 cung cấp giải pháp mạng Wi-Fi hoàn chỉnh và khép kín, cho phép nó có thể lưu trữ các ứng dụng hoặc để giảm tải chức năng mạng Wi-Fi của một hệ thống.

Nó có tích hợp bộ nhớ cache để cải thiện hiệu suất của hệ thống; phục vụ như một Wi-Fi Adapter, có thể truy cập internet không dây qua UART hoặc CPU qua cầu AHB (Advanced High-performance Interface).

Khả năng xử lý on-board và lưu trữ cho phép nó được tích hợp với các bộ cảm biến và các thiết bị cụ thể, ứng dụng khác thông qua GPIOs.

Hệ thống phức tạp được thiết kế để sleep / wake up cho VoIP energyefficient, radio, thích ứng cho hoạt động năng lượng thấp, cho các ứng dụng có sử dụng Bluetooth, DDR, LVDS, và hỗ trợ giảm nhiễu LCD.

ESP8266 đã được thiết kế cho di động, thiết bị điện tử và các ứng dụng điều khiển sử dụng Internet, với mục đích đạt được mức tiêu thụ điện năng thấp nhất sử dụng một số kỹ thuật độc quyền. Các kiến ​​trúc tiết kiệm năng lượng hoạt động ở 3 chế độ: chế độ hoạt động, ngủ và chế độ ngủ sâu

Bằng cách sử dụng các kỹ thuật quản lý nguồn, giảm tải không cần thiết. Giữa mode ngủ và mode hoạt động, ESP8266 tiêu thụ ít hơn 12uA ở chế độ ngủ và ít hơn 1.0mW (DTIM = 3) hoặc ít hơn 0.5mW (DTIM = 10) để ở mode hoạt động.

Khi ở chế độ ngủ, chỉ có đồng hồ thời gian thực và cơ quan giám sát vẫn hoạt động. Đồng hồ thời gian thực có thể được lập trình để đánh thức những ESP8266 ở bất kỳ khoảng thời gian cần. ESP8266 có thể được lập trình để thức dậy khi một điều kiện cụ thể được phát hiện.

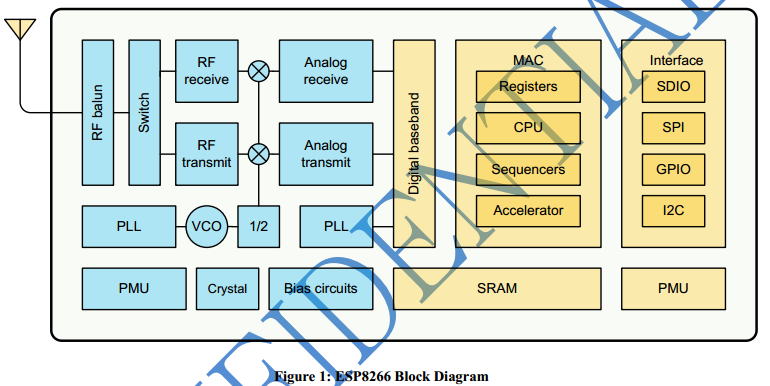
Trong project này, ESP8266-01 được sử dụng làm module Wifi:

* Sử dụng UART để giao tiếp

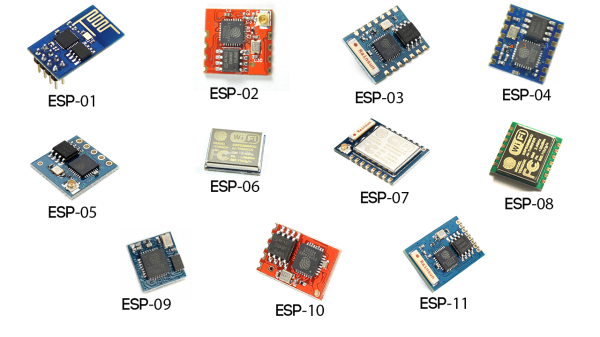
Để truyền nhận được với Atmega16 trong project này thì khung truyền phải được thiết lập cùng với các thông số tương ứng như của Atmega16

* Sử dụng Wifi để tạo mạng và Server lắng nghe
* Sử dụng Firmware NodeLua

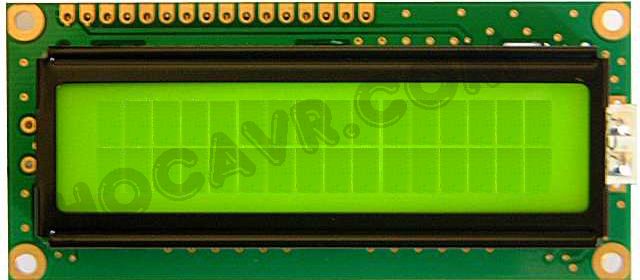
NodeLua là một firmware đa nền tảng dựa trên ngôn ngữ Lua giúp lập trình dễ dàng hơn. Là một trình thông dịch, với các Cloud API. Nó hiện chạy trên ESP8266 và đang phát triển trên các chip khác, dựa trên ngôn ngữ Lua, giống như Nodejs API, nhưng nhanh hơn 10 lần và nhỏ hơn 100 lần so với Nodejs, phù hợp với module với chỉ một vài trăm kilobyte bộ nhớ Flash/RAM.



Họ ESP8266 (ESP8266 Family)



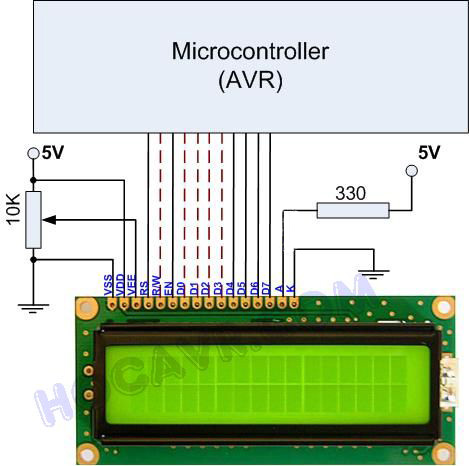
## Text LCD

Text LCD là các loại màn hình tinh thể lỏng nhỏ dùng để hiển thị các dòng chữ hoặc số trong bảng mã ASCII. Text LCD được chia sẵn thành từng ô và ứng với mỗi ô chỉ có thể hiển thị một ký tự ASCII; các mẫu ký tự được định nghĩa sẵn.

Text LCD có 2 cách giao tiếp cơ bản là nối tiếp (như I2C) và song song. Trong project này sử dụng loại giao tiếp song song, cụ thể là LCD 16x2 điều khiển bởi chip HD44780U của hãng Hitachi.

HD44780U là bộ điều khiển cho các Text LCD dạng ma trận điểm (dot-matrix), chip này có thể được dùng cho các LCD có 1 hoặc 2 dòng hiển thị. HD44780U có 2 mode giao tiếp là 4 bit và 8 bit. Nó chứa sẵn 208 ký tự mẫu kích thước font 5x8 và 32 ký tự mẫu font 5x10 (tổng cộng là 240 ký tự mẫu khác nhau).

Project này sử dụng mode điều khiển 4 bit. Trong chế độ này, các chân D0, D1, D2 và D3 của LCD không được sử dụng (để trống), chỉ có 4 chân từ D4 đến D7 được kết  nối với chip bộ điều khiển ngoài.

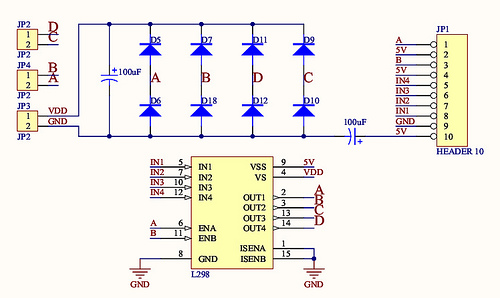
Sơ đồ chân:

* Vss (GND): Ground
* Vdd (VCC): Nguồn cho LCD
* Vee: Chỉnh độ tương phản
* RS: Điều khiển LCD theo chế độ ở D0-7 là lệnh hay dữ liệu
* R/W: Điều khiển LCD theo chế độ Ghi(từ AVR vào LCD)/Đọc(từ LCD vào AVR)
* E: điều khiển LCD vô hiệu hóa LCD/LCD hoạt động/Bắt đầu ghi/đọc LCD
* D0-7: truyền nhận dữ liệu/lệnh
* Trong một số LCD 2 chân LED nền được đánh số 15 và 16 nhưng trong một số trường hợp 2 chân này được ghi là A (Anode) và K (Cathode).

Trong project này, LCD được sử dụng để test và hiển thị các thông tin cần thiết ngay trên xe.

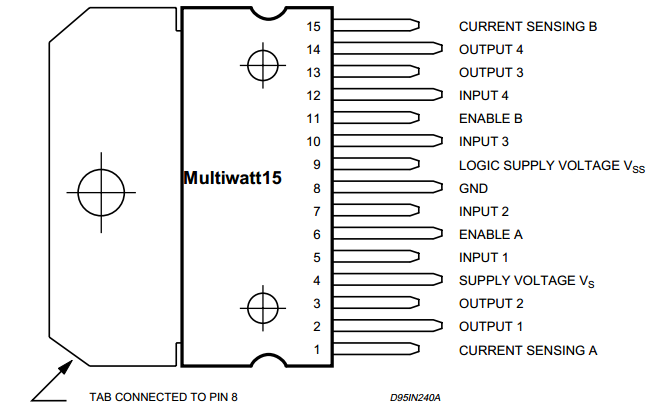
## Module L298

IC L298 là một IC tích hợp nguyên khối gồm 2 mạch cầu H bên trong. Với điện áp làm tăng công suất đầu ra từ 5V – 47V , dòng lên đến 4A, L298 rất thích hợp trong những ứng dụng công suất nhỏ như động cơ DC loại vừa …



Sơ đồ chân

* 4 chân INPUT: IN1, IN2, IN3, IN4 được nối lần lượt với các chân 5, 7, 10, 12 của L298. Đây là các chân nhận tín hiệu điều khiển.
* 4 chân OUTPUT: OUT1, OUT2, OUT3, OUT4 (tương ứng với các chân INPUT) được nối với các chân 2, 3, 13, 14 của L298. Các chân này sẽ được nối với động cơ.
* Hai chân ENA và ENB dung để điều khiển các mạch cầu H trong L298. Nếu ở mức logic “1” (nối với nguồn 5V) thì cho phép mạch cầu H hoạt động, nếu ở mức logic “0” thì mạch cầu H không hoạt động.



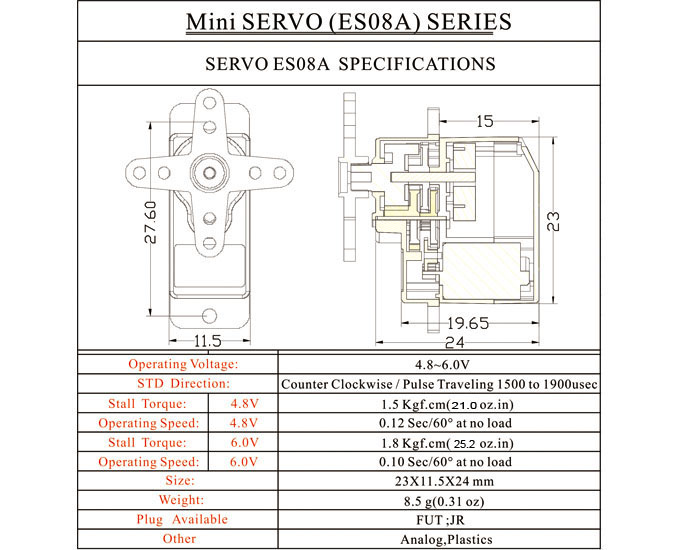
Trong project này, IC L298 đã được tích hợp trong module L298, được sử dụng để dẫn xung PWM từ VĐK ra tới động cơ, nhằm mục đích cách ly dòng sử dụng của động cơ.

## Mini Servo ES08A

Servo EMAX ES08 là mini servo cho người mới bắt đầu, bao gồm đầy đủ các trục phổ biến, thường được dùng trong các dự án cơ điện tử nhỏ.

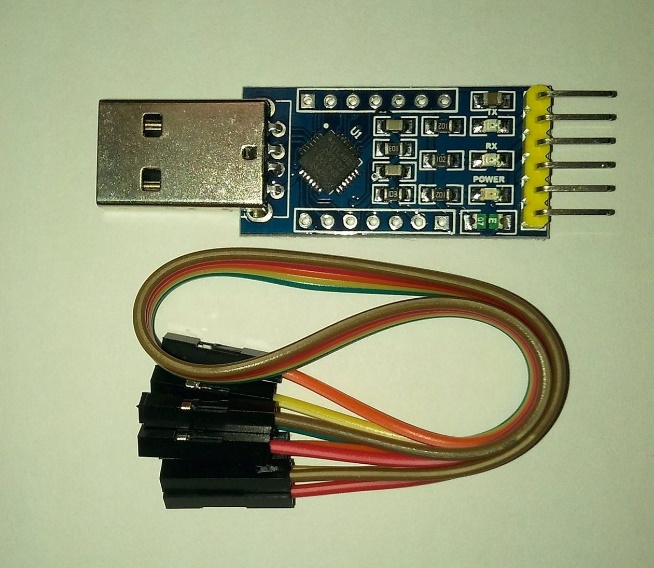
Servo gồm một tổ hợp gồm 1 động cơ DC công suất nhỏ, hộp giảm tốc và bộ điều khiển góc quay. Có 2 loại chính là Servo thường và digital Servo, EMAX ES08 là Servo thường.

Servo thường có 3 dây, dây màu đen là dây GND, dây đỏ là dây nguồn (thường là 5V) và 1 dây trắng hoặc vàng và dây điều khiển. Vì các Servo đã có sẵn mạch điều khiển góc quay bên trong nên chúng ta không cần bất cứ giải thuật gì mà chỉ cần cấp tín hiệu PWM cho dây điều khiển là Servo có thể xoay đến 1 vị trí nào đó (chú ý là Servo thường chỉ xoay nửa vòng, điều khiển servo là điều khiển góc xoay chứ không phải điều khiển cận tốc xoay).



Trong project này, servo ES08 được sử dụng để điều khiển hệ thống bánh lái trước cho xe.

## Module USB to COM CP2102

USB to TTL UART CP2102 là module chuyển đổi tín hiệu USB sang tín hiệu Serial tuần tự theo chuẩn TTL. Đây là module của Silicon, dùng để giao tiếp giữa cổng USB trên máy tính và UART. Module này sử dụng chip CP2102, hỗ trợ các hệ điều hành Win 8, Win7, Vista, Server 2003, XP, Mac OS - X và các hệ điều hành phổ biến khác.

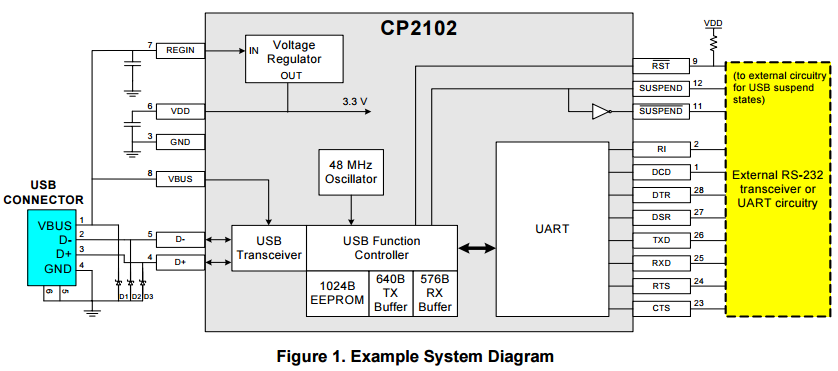
CP2102 hoạt động ở mức áp 3.3V nên có thể dùng để giao tiếp Serial với các IC chạy ở mức 3.3V như các router wifi, Raspberry Pi, bluetooth module...

CP2102 có kích thước nhỏ gọn và yêu cầu rất ít thành phần bên ngoài để hoạt động được ngay. CP2102 không sử dụng thạch anh ngoài như các chip PL2303.

Module có sẵn ngõ ra điện áp 3.3V với dòng tải tối đa là 100mA.

Trên mạch có 5 cổng đầu ra: 3.3V, RST, 5V, Tx, Rx, Gnd. Trong đó chân RST được sử dụng để ngắt kết nối USB của module với máy tính.

Tốc độ tối đa 115200 bps.



Trong project này Module CP2102 dùng để kêt nối Module ESP8266 vào máy tính qua USB để nạp code.

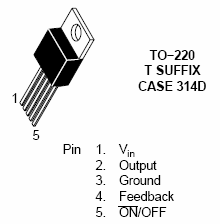
## IC LM2576

Module nguồn trong project này sử dụng 2 IC LM2576 với output 5V và 3.3V

LM2576 là một IC nguồn với dòng điện định mức đầu ra tải là 3A và có các đầu điện áp đầu ra cố định 3.3V, 5V, 12V, 15V và điện áp biến đổi tùy từng loại Serial của LM2576. Đây là loại IC nguồn cung cấp điện áp đầu ra ổn định, hoạt động ổn định với đầu tản nhiệt tốt giúp IC hoạt động tốt trong nhiệt độ cho phép.

Một số tính năng chính của LM2576

* Áp Output là 3.3V, 5V, 12V, 15V và loại điều chỉnh được
* Điện áp điều chỉnh được từ 1.25V đến 37V
* Dòng output định mức là 3A
* Áp input định mức là 40V có thể lớn 60V tùy từng dòng Serial
* Tần số đóng cắt chuẩn 52Khz
* Hiệu suất cao
* Bảo vệ quá dòng và quá nhiệt

Sơ đồ chân

* Chân 1 (Vin) : Chân nguồn đầu vào
* Chân 2 (Vout) : Chân điện áp đầu ra
* Chân 3 (GND) : Chân nguồn chung
* Chân 4 (Feedback) : Chân đưa tín hiệu phản hồi từ đầu ra về đầu vào. Đưa vào bộ so sánh để điều chỉnh ổn định điện áp
* Chân 5 (On/Off) : Chân đóng mở. Thường để GND

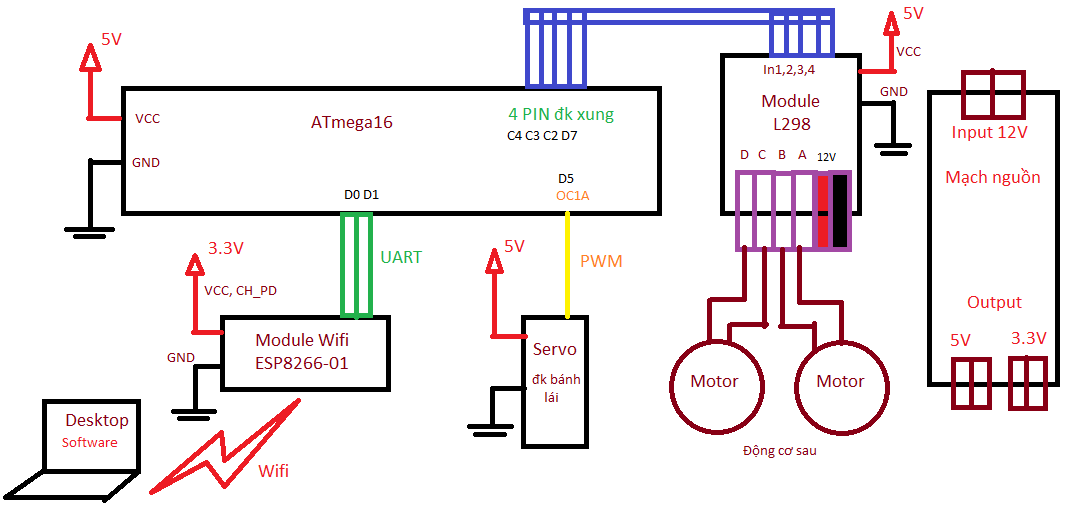
Thông số chính của LM2576

* Điện áp đầu vào : LM2576 : 45V
* Dòng điện đầu ra : 3A
* Giải nhiệt độ hoạt động là : −65˚C to +150˚C

# PHẦN 2: PHÂN TÍCH THIẾT KẾ

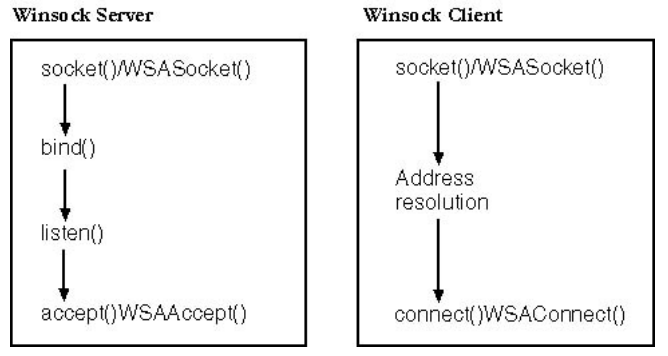
## Mô hình ghép nối và liên kết

### Mô hình ghép nối phần cứng



* Mạch nguồn:
  + Input 12V
  + Output 5V, 3.3V
* Atmega16:
  + Lấy nguồn 5V từ mạch nguồn
  + 4 pin C4, C3, C2, D7 output tín hiệu xung ra Module L298 để điều khiển động cơ
  + 2 pin D0 (RxD), D1 (TxD) giao tiếp UART với Module Wifi để nhận lệnh và trả lại dữ liệu yêu cầu
  + Pin D5 (OC1A) xuất xung PWM điều khiển Servo (Bánh lái của xe)
* Module L298:
  + Lấy nguồn 5V từ mạch nguồn cho chip L298
  + Lấy nguồn 12V từ mạch nguồn cho động cơ
  + 4 pin In1, In2, In3, In4 nhận xung điều khiển từ Atmega16
  + 4 pin A, B, C, D xuất điện áp tương ứng với xung điều khiển ra 2 động cơ
* Servo:
  + Lấy nguồn 5V từ mạch nguồn
  + Pin nhận tín hiệu PWM từ Atmega16 để điều khiển bánh lái
* Module Wifi:
  + Lấy nguồn 3.3V từ mạch nguồn
  + 2 pin RxD, TxD giao tiếp UART với Atmega16 để truyền lệnh và dữ liệu cần thiết
  + Server lắng nghe ở IP 192.168.4.1, Subnet Mask 255.255.255.0, Gateway 192.168.4.1, Port 80
* Desktop:
  + Kết nối vào mạng Wifi do Module Wifi phát ra
  + Là client kết nối với Server ở địa chỉ trên, truyền nhận lệnh và dữ liệu cho Atmega16 thông qua Module Wifi

### Mô hình phần mềm điều khiển



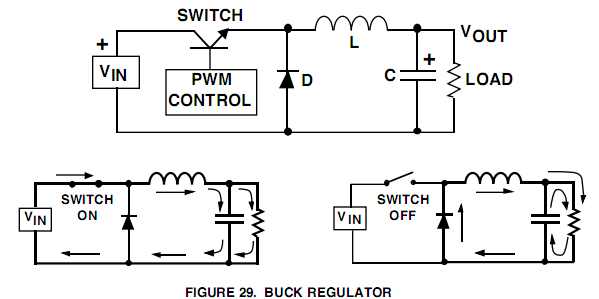
Mô hình mạng

Phần mềm sử dụng mô hình mạng Client – Server với Server được cài đặt trên Module Wifi, còn Desktop là Client, Client gửi lệnh điều khiển đến Server

### Mạch nguồn

LM2576 hoạt động theo nguyên lý mạch Buck, lm2576 đóng vai trò như switch đóng ngắt, tạo xung để xuất áp tương ứng, và bộ so sánh với feedback để điều chỉnh áp ra chuẩn hơn. Nguyên lý mạch Buck là kiểu biến đổi nguồn cho điện áp đầu ra nhỏ hơn so với điện áp đầu vào tức là *Vin < Vout*

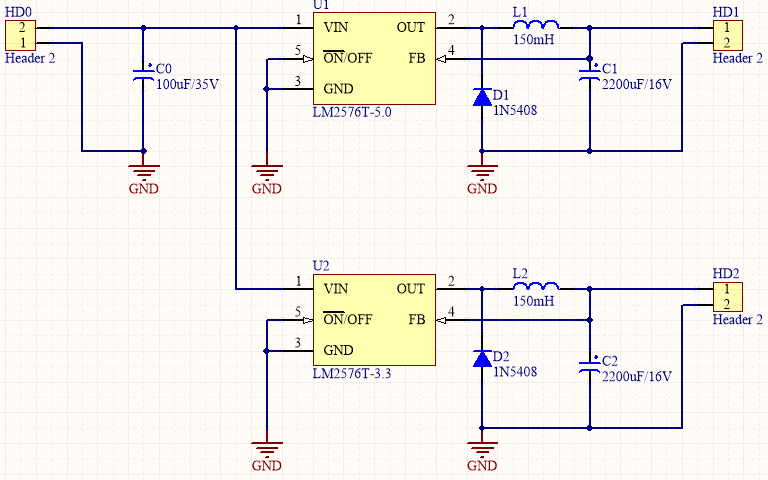
Xét một mạch nguyên lý sau :



Mạch chỉ dùng một van đóng cắt nguồn điện và phần lọc đầu ra. Điện áp đầu ra được điều biến theo độ rộng xung

* Khi " Switch On" thì dòng điện đi qua cuộn cảm và dòng điện trong cuộn cảm tăng lên, tại thời điểm này thì tụ điện được nạp đồng thời cũng cung cấp dòng điện qua tải.
* Khi " Swith Off", trong cuộn cảm tích lũy năng lượng từ trường và tụ điện điện được tích lũy trước đó sẽ phóng qua tải. Cuộn cảm có xu hướng giữ cho dòng điện không đổi và giảm dần.
* Quá trình đóng cắt liên tục tạo tải một điện áp trung bình theo luật băm xung PWM. Dòng điện qua tải sẽ ở dạng xung tam giác đảm bảo cho dòng liên tục qua tải. Tần số đóng cắt khá cao để đảm bảo triệt nhiễu công suất cho mạch.

Dưới đây là thiết kế mạch nguồn trong project này



Nguyên lý giống như nguyên lý mạch Buck.

* Tụ C0 để lọc nhiễu cho áp vào
* Diode 1N5408 để chống dòng ngược, hạn dòng tối đa 3A, bảo vệ cho IC LM2576. Theo nguyên lý Buck, diode 1N5408 để cung cấp đường dẫn dòng đi vào cuộn cảm, khi Switch (ở đây là xung phát ra bởi LM2576) off.

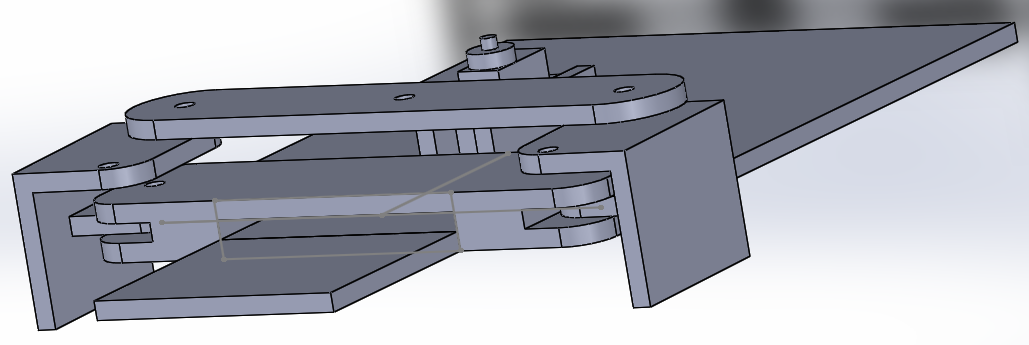
### Thiết kế cơ khí xe

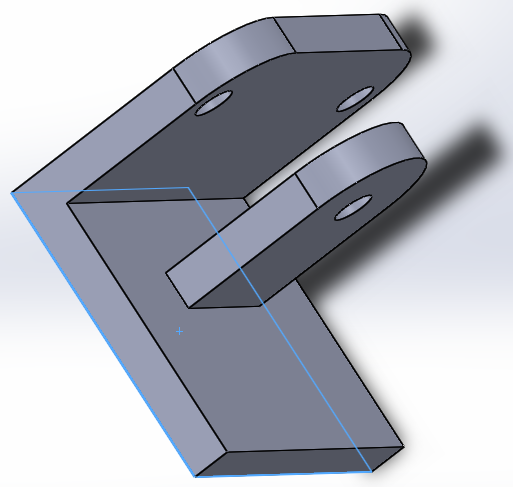
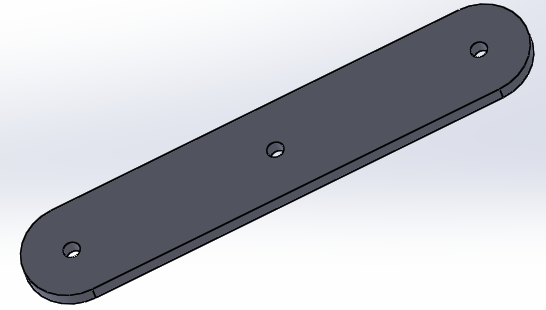
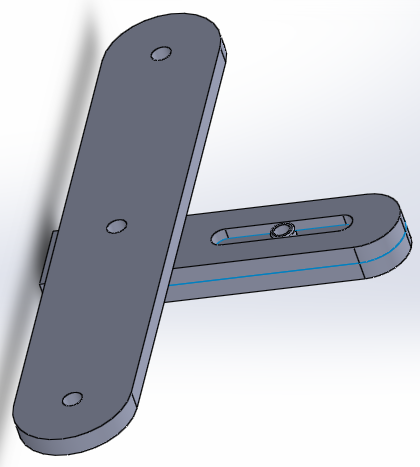
Dưới đây là thiết kế tổng quan cho bộ phận lái, bộ phận động cơ không thiết kế

Thông số chi tiết nằm trong file SolidWorks kèm báo cáo

Thông số thực tế có thể sai khác so với bản thiết để phù hợp với thực tế

Cơ cấu bánh lái tổng quan



* Khớp quay:
  + Bánh lái trước cố định vào khớp quay
* Thanh đánh hướng:
  + Được điều hướng bằng tay đánh hướng
  + Có 3 trục, trục ở tâm gắn vào tay đánh hướng, 2 trục 2 bên gắn vào trục quay của khớp quay
* Tay đánh hướng (sau khi gắn thanh đánh hướng vào)
  + Có gắn bộ phận bánh răng của servo

## Thông số kỹ thuật

* Atmega16
  + f = 8MHz
  + Khởi động module UART, Interrupt, Timer/Counter1, Text LCD 16 bit
  + Set Output, Input cho các Port cần thiết
  + 4 Pin C4, C3, C2, D7 dùng cho điều khiển chiều quay của động cơ
* Module Wifi
  + f = 80MHz (mặc định – ngoài ra chỉ có thể là 160MHz)
  + Khởi động chế độ wifi có thể tạo được mạng LAN và kết nối tới Wifi router khác đồng thời
  + Khởi động module UART
* Wifi
  + Khởi tạo server trên Module wifi
  + Server lắng nghe trên cả mạng do module Wifi tạo ra và mạng kết nối với Router Wifi khác
  + Giao thức TCP, lắng nghe ở Port 80
* UART
  + Baud: 9600
  + Databits: 8
  + Parity: None
  + Stopbits: 1
* PWM
  + Do Timer/Counter1 của Atmega16 tạo ra
  + f = 1MHz (tự sinh bộ chia tần số Atmega16 cho 8)
  + Chọn mode 14: Fast PWM
  + Chọn dạng tín hiệu ra của PWM là Clear khi gặp giá trị trong OCR1A, và Set khi đạt TOP trong ICR1 (= 20000)
  + OCR1A được chọn sao cho góc quay của bánh lái quay 1 góc hợp lý với phần cơ khí cho xe hướng thẳng, quay trái hoặc phải
* Packet lưu data
  + Mode 0: 3 byte
    - Byte 0: = ‘0’
    - Byte 1: = ‘0’, ‘1’, ‘2’: hướng quay của bánh lái
    - Byte 2: = ‘0’, ‘1’, ‘2’: hướng đi của bánh sau
  + Mode 1: 3 byte: bỏ qua 2 bánh lái, gắn thêm bánh tự lái
    - Byte 0: = ‘1’
    - Byte 1: không quan tâm
    - Byte 2: = ‘0’, ‘1’, ‘2’, ‘3’, ‘4’: hướng đi hoặc quay của bánh sau
  + Mode 2: số byte không xác định, vừa nhận data vừa xử lý
    - Byte 0: = ‘2’
    - Byte 1, 2: là 2 số có 1 chữ số, là số byte của độ dài của địa chỉ IP và Subnet Mask
    - Byte xx, yy: là 2 số chỉ có thể là số có 1 hoặc 2 chữ số, là độ dài địa chỉ IP và Subnet Mask
    - Byte x..x, y..x: là chuỗi byte địa chỉ IP và Subnet Mask
  + Mode 3: Dự trữ
  + Mode 4: gửi packet ‘400’: yêu cầu ngắt kết nối từ phía Server

## Các hàm chức năng

### Atmega16

* void Exec\_Cmd(int mode)

Xử lý các byte data nhận được, xử lý dịch lệnh tương ứng, từ đó dùng các hàm chức năng điều khiển các cơ cấu chấp hành

* + mode: byte đầu tiên được gửi đến, chọn mode cho hệ thống

Hàm được gọi ngay sau khi byte đầu tiên trong Packet nhận được

* void Control\_mode0(int dir, int state)

Điều khiển mode 0 của xe

* + dir: byte thứ 2 được gửi của mode 0, chọn hướng cho bánh lái
  + state: byte thứ 3 được gửi của mode 0, chọn chiều đi cho xe

Hàm được gọi sau khi hàm Exec\_Cmd xử lý được mode sẽ dùng cho xe

* void Control\_mode1(int state)

Điều khiển mode 1 của xe

* + state: byte thứ 3 được gửi của mode 1 (byte thứ 2 nhận nhưng không quan tâm), chọn chiều đi hoặc quay tại chỗ cho xe

Hàm được gọi sau khi hàm Exec\_Cmd xử lý được mode sẽ dùng cho xe

* void Control\_mode2()

Điều khiển mode 2 của xe. Mode 2 là mode gửi lệnh đến Module wifi qua UART, yêu cầu Module wifi trả về IP Address và Subnet Mask của Server và hiển thị lên Text LCD trên xe

Hàm được gọi sau khi hàm Exec\_Cmd xử lý được mode sẽ dùng cho xe

* interrupt [EXT\_INT1] void ext\_int1\_isr(void)

Lệnh cho Atmega16 gửi dữ liệu mang cú pháp lệnh đến Module wifi qua UART, yêu cầu Module wifi gửi trả về IP Address và Subnet Mask của Server và hiển thị lên Text LCD trên xe

### Module ESP8266

* wifi.setmode(mode)

Set chế độ cho module wifi, có các mode sau:

* + wifi.STATION: chỉ kết nối tới wifi router khác
  + wifi.SOFTAP: module trở thành 1 Access Point
  + wifi.STATIONAP: cả 2 chế độ trên

Hàm được gọi khi khởi động module wifi

* wifi.ap.config(cfg)

Set SSID và PASSWORD cho mạng cục bộ do module wifi tạo ra, có thể chứa các tham số:

* + string: SSID 1-32 kí tự
  + string: pwd 8-64 kí tự
  + int: kiểu mã hóa AUTH\_OPEN | AUTH\_WPA\_PSK | AUTH\_WPA2\_PSK | AUTH\_WPA\_WPA2\_PSK default=Open
  + int: kênh 1-14 default = 6
  + int: ẩn = 1; không ẩn = 0; default = 0
  + int: số kết nối tối đa 1-4 default = 4
  + int: chu kì của beacon 100-60000 default = 100

(Beacon là một dạng frame ngắn được gởi từ AP đến các trạm client (trong mạng Infrastructure) hoặc từ trạm đến trạm (trong mạng Ad-Hoc) để tổ chức và đồng bộ hóa các truyền thông trong mạng WLAN)

Hàm được gọi khi khởi động module wifi

* uart.setup(id, baud, databits, parity, stopbits, echo)

Set chế độ hoạt động cho module UART trong ESP8266

* + id = 0, chỉ cho phép 1 uart
  + baud = 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 74880, 115200, 230400, 460800, 921600, 1843200, 2686400.
  + databits = 5, 6, 7, 8.
  + parity = 0 (none), PARITY\_ODD, PARITY\_EVEN.
  + stopbits = 1(1 stopbit), STOPBITS\_1, STOPBITS\_1\_5, STOPBITS\_2.
  + echo = 0 (tắt echo back).

Hàm được gọi khi khởi động module wifi

* net.createServer(type, timeout)

Khởi tạo server trên module wifi

* + type: net.TCP hoặc net.UDP
  + timeout: cho TCP server, timeout khoảng 1~28800 seconds, để ngắt kết nối với client không hoạt động

Hàm được gọi khi khởi động module wifi

* net.server.listen(port,[ip],function(net.socket))

Set chế độ cho Server lắng nghe

* + port: số port
  + ip: địa chỉ ip, có thể bỏ qua
  + function(net.socket): hàm callback, xảy ra khi 1 kết nối thành công

Hàm được gọi khi khởi động module wifi

* net.socket:on(event, function cb())

Khai báo hàm callback cho sự kiện xảy ra

* + event: string, có thể là: "connection", "reconnection", "disconnection", "receive", "sent"
  + function cb(net.socket, [string]): hàm callback, tham số đầu tiên là socket; nếu event là "receive" thì tham số thứ 2 là buffer để lưu data

Hàm được khai báo trong hàm callback của hàm listen ở trên

* trans\_ip\_ap()

Get IP Address và Subnet Mask của Module Wifi, đồng thời gửi về cho Atmega16

Hàm được gọi khi nhận được data chứa byte đầu tiên bằng ‘2’.

### Phần mềm điều khiển

* public void ConnectServer()

Là nội dung Thread kết nối đến IP, Port của Server. Sau khi có kết nối, gọi Thread liên tục lắng nghe data, rồi hủy ngay chính nó để ngưng tìm kiếm Server

Hàm được gọi sau khi Click nút Connect

* public void ListenData(object obj)

Là nội dung Thread lắng nghe dữ liệu liên tục từ server

* + kết nối đến server

Hàm được gọi ngay khi có kết nối đến Server

* public bool IsConnect(Socket sk)

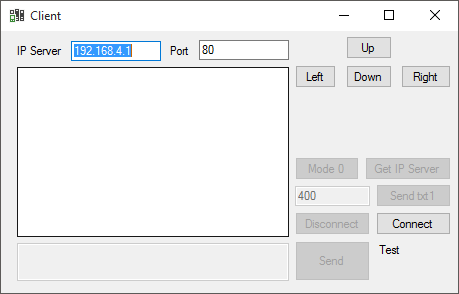
Kiểm tra socket có kết nối không với Timeout 1000us

* + sk: socket chứa kết nối đến Server

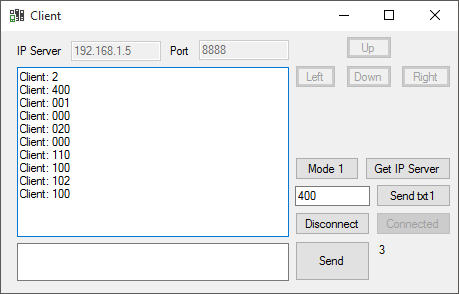
Hàm được gọi mỗi trước khi thực hiện một thao tác gửi nhận dữ liệu liên quan đến socket kết nối đến Server

# PHẦN 3: SẢN PHẨM KẾT QUẢ

## Giao diện phần mềm điều khiển



* Chọn IP Server và Port
* Click button Connect để tìm kiếm Server và kết nối
* Nếu có kết nối thì các button và text box sẽ được bật để sử dụng



* Click button Mode để chọn Mode 0 hoặc 1
* Click Get IP Server để chọn Mode 2
* Click chuột vào ô text box hiển thị dữ liệu đi và đến để focus vào đó, sau đó sử dụng 4 nút mũi tên trên bàn phím để điều khiển xe theo ý muốn
* Click Disconnect để ngắt kết nối đến Server từ phía Client
* Click Send txt 1 để gửi Packet ‘400’ để yêu cầu ngắt kết nối từ phía Server

## Sản phẩm Xe

