# THÔNG TIN PROJECT

**Ô TÔ ĐIỀU KHIỂN TỪ XA BẰNG PHẦN MỀM DI ĐỘNG HOẶC GIỌNG NÓI QUA WIFI DI ĐỘNG**

**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 |

**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/

**MỤC LỤC**

[THÔNG TIN PROJECT 1](#_Toc441276167)

[Tài liệu tham khảo 1](#_Toc441276168)

[MỤC LỤC 2](#_Toc441276169)

[PHẦN I: NỀN TẢNG VÀ CÔNG NGHỆ PHÁT TRIỂN ỨNG DỤNG 3](#_Toc441276170)

[1.1. Công nghệ kỹ thuật hỗ trợ liên quan 3](#_Toc441276171)

[1.1.1. Microsoft Visual Studio 3](#_Toc441276172)

[1.1.2. CodevisionAVR 3](#_Toc441276173)

[1.1.3. Progisp 3](#_Toc441276174)

[1.1.4. Nodemcu Firmware Programmer 3](#_Toc441276175)

[1.1.5. ESPlorer 3](#_Toc441276176)

[1.1.6. Altium Designer 3](#_Toc441276177)

[1.1.7. SolidWorks 3](#_Toc441276178)

[1.2. ATmega16 4](#_Toc441276179)

[1.3. Module ESP8266 5](#_Toc441276180)

[1.4. Text LCD 7](#_Toc441276181)

[1.5. Module L298 7](#_Toc441276182)

[1.6. Mini Servo ES08A 8](#_Toc441276183)

[1.7. Module USB to COM CP2102 8](#_Toc441276184)

[1.8. IC LM2576 9](#_Toc441276185)

[PHẦN 2: PHÂN TÍCH THIẾT KẾ 10](#_Toc441276186)

[2.1. Mô hình ghép nối và liên kết 10](#_Toc441276187)

[2.1.1. Mô hình ghép nối phần cứng 10](#_Toc441276188)

[2.1.2. Mô hình phần mềm điều khiển 11](#_Toc441276189)

[2.1.3. Mạch nguồn 11](#_Toc441276190)

[2.1.4. Thiết kế cơ khí xe 12](#_Toc441276191)

[2.2. Thông số kỹ thuật 14](#_Toc441276192)

[2.3. Các hàm chức năng 15](#_Toc441276193)

[2.3.1. Atmega16 15](#_Toc441276194)

[2.3.2. Module ESP8266 15](#_Toc441276195)

[2.3.3. Phần mềm điều khiển trên máy tính 17](#_Toc441276196)

[2.3.4. Phần mềm điều khiển trên di động 17](#_Toc441276197)

[PHẦN 3: SẢN PHẨM KẾT QUẢ 18](#_Toc441276198)

[3.1. Giao diện phần mềm điều khiển trên máy tính 18](#_Toc441276199)

[3.2. Giao diện phần mềm điều khiển trên di động 19](#_Toc441276200)

[3.3. Sản phẩm Xe 19](#_Toc441276201)

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

## Công nghệ kỹ thuật hỗ trợ liên quan

### Microsoft Visual Studio

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

### CodevisionAVR

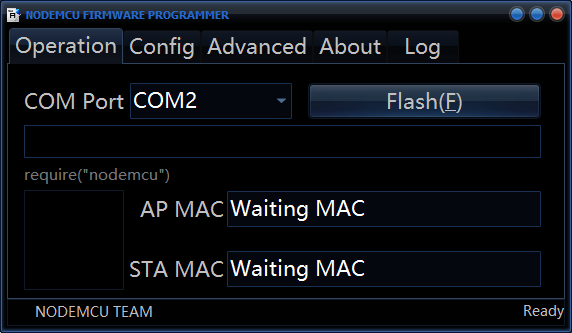
Trong project này, CodeVision được dùng để lập trình cho VĐK với ngôn ngữ C.

### 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

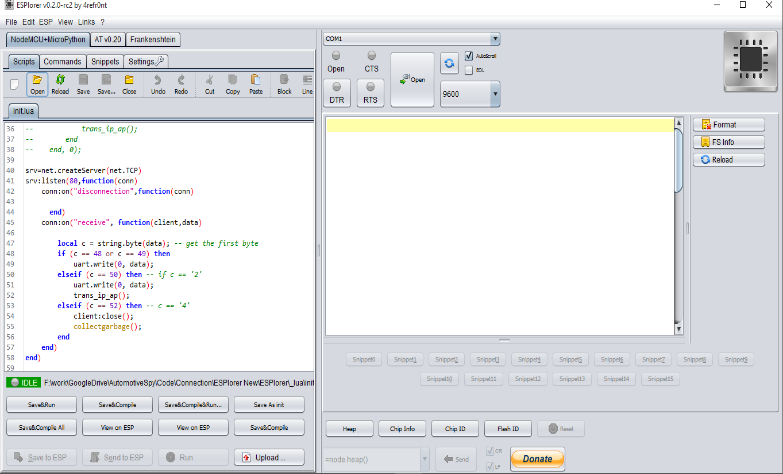
Trong project này, Progisp được dùng để nạp code cho ATmega16

### 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.

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

### SolidWorks

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

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

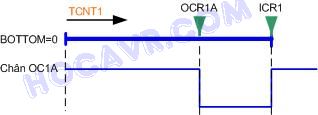
T/C là các module độc lập với CPU. Chức năng chính của các bộ T/C, như tên gọi của chúng, là định thì (tạo ra một khoảng thời gian, đếm thời gian…) và đếm sự kiện. Trên các chip AVR, các bộ T/C còn có thêm chức năng tạo ra các xung điều rộng PWM (Pulse Width Modulation), ở một số dòng AVR, một số T/C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

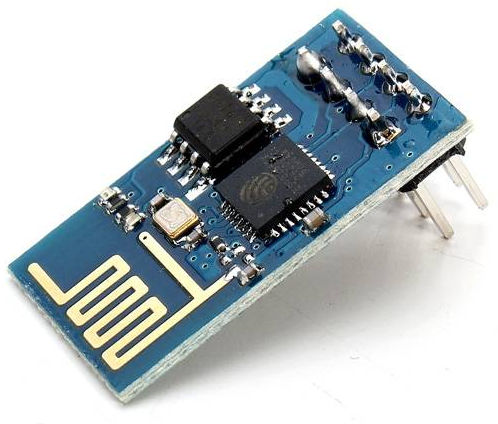
T/C1: là bộ định thời, đếm đa năng 16 bit. Bộ T/C này có 5 chế độ hoạt động chính. Ngoài các chức năng thông thường, T/C1 còn được dùng để tạo ra xung điều rộng PWM dùng cho các mục đích điều khiển. Có thể tạo 2 tín hiệu PWM độc lập trên các chân OC1A (chân 15) và OC1B (chân 16) bằng T/C1.

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

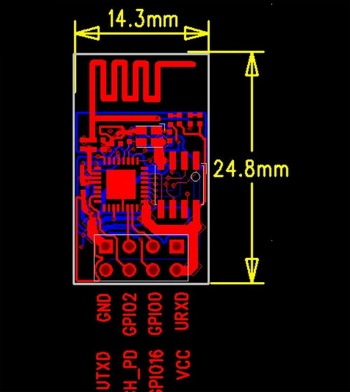
## Module ESP8266

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.

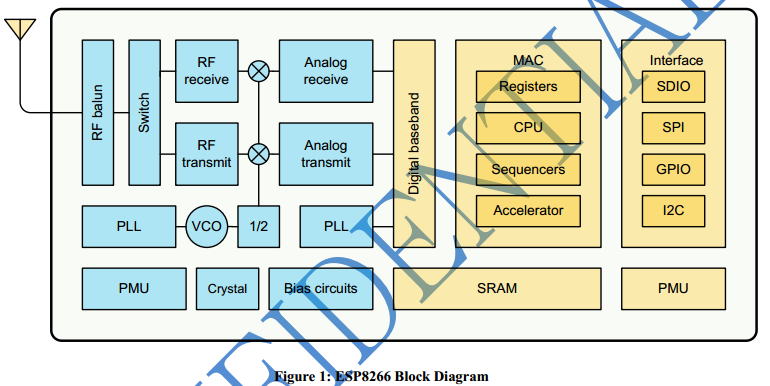
Trong project này, ESP8266-01 được sử dụng làm module Wifi:

* Sử dụng UART để giao tiếp

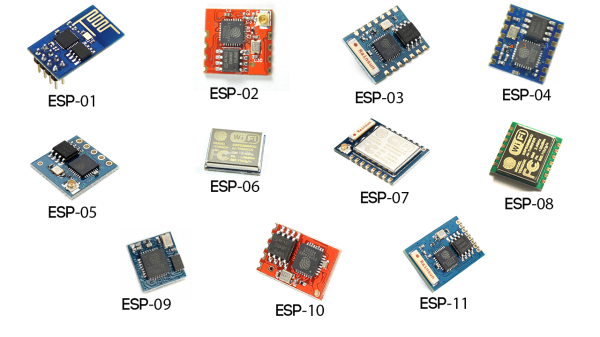
Để truyền nhận được với Atmega16 trong project này thì khung truyền phải được thiết lập cùng với các thông số tương ứng như của Atmega16

* 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)

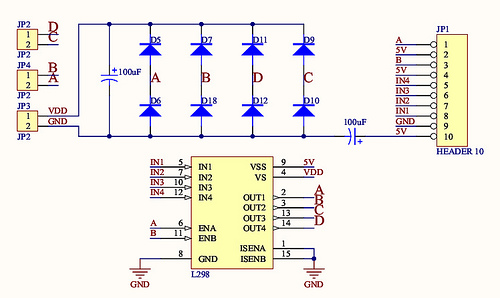


## Text LCD

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

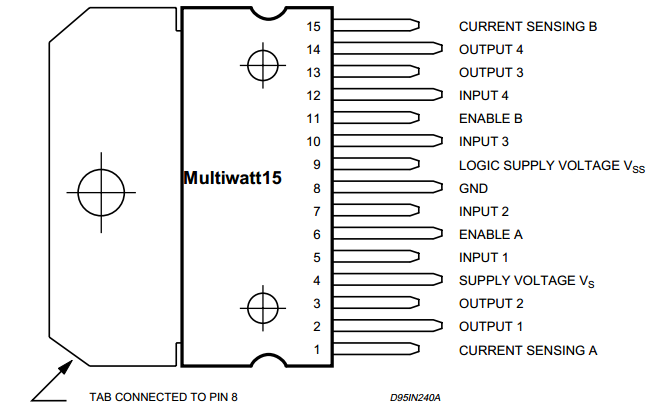
## Module L298

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

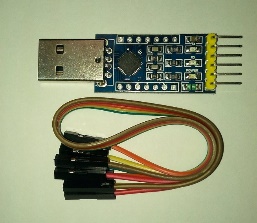
* 4 chân INPUT: IN1, IN2, IN3, IN4 được nối lần lượt với các chân 5, 7, 10, 12 của L298. Đây là các chân nhận tín hiệu điều khiển.
* 4 chân OUTPUT: OUT1, OUT2, OUT3, OUT4 (tương ứng với các chân INPUT) được nối với các chân 2, 3, 13, 14 của L298. Các chân này sẽ được nối với động cơ.
* Hai chân ENA và 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.

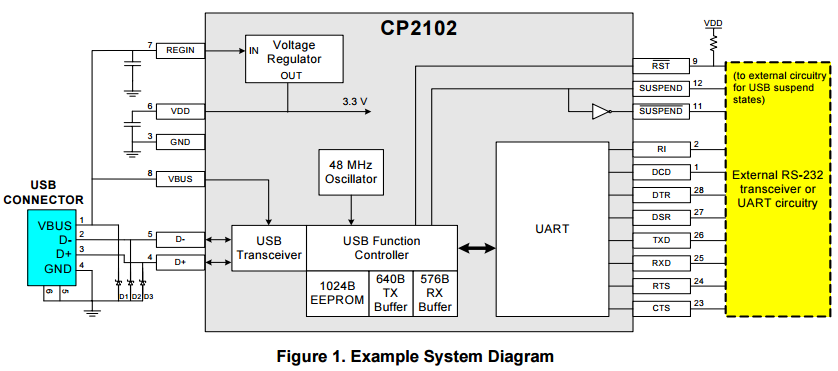
## Mini Servo ES08A

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



## Module USB to COM CP2102

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.

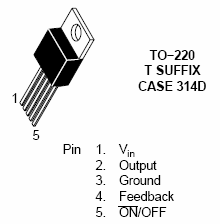
## IC LM2576

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

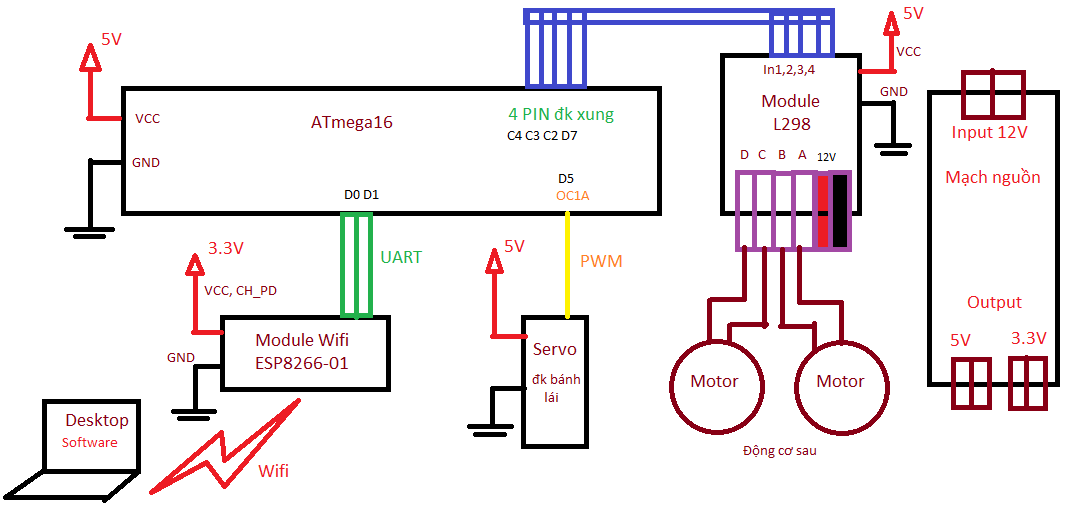
Thông số chính của LM2576

* 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

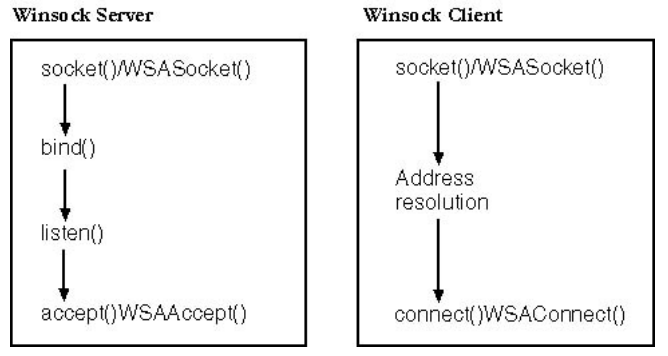
## Mô hình ghép nối và liên kết

### 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



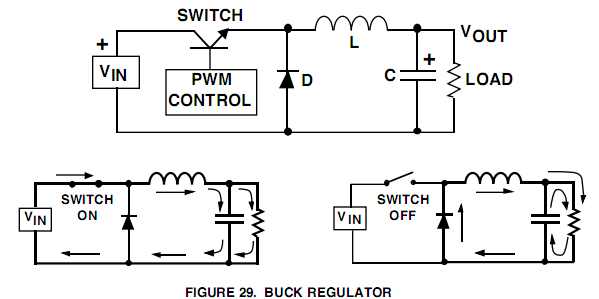
Mô hình mạng

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*

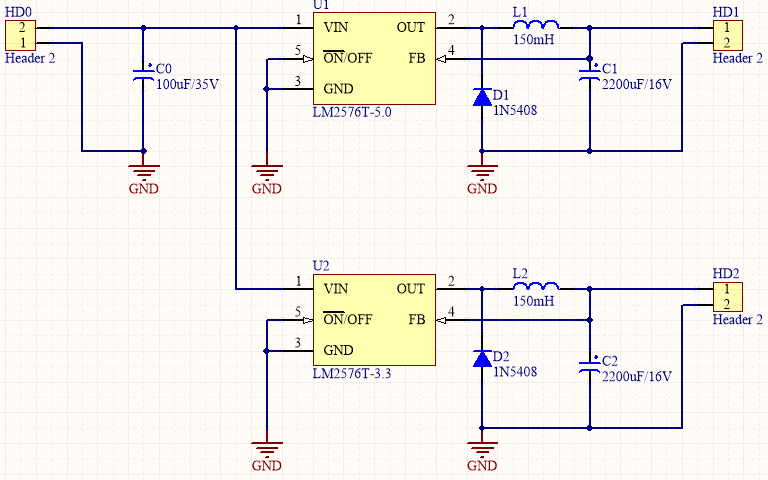
Xét một mạch nguyên lý sau :



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.

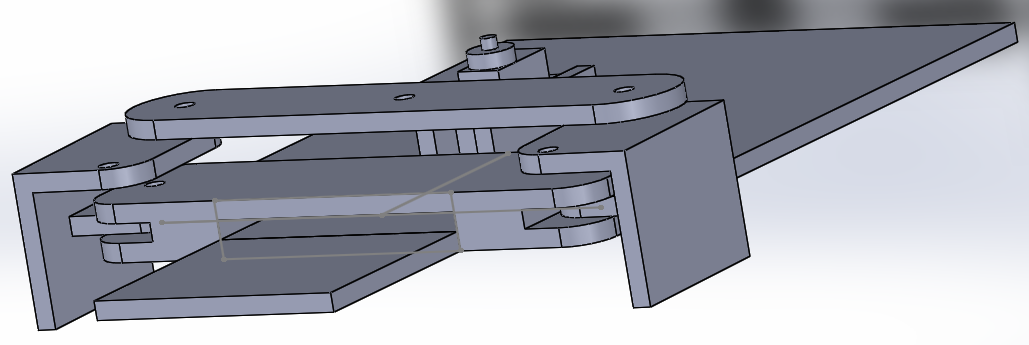
### Thiết kế cơ khí xe

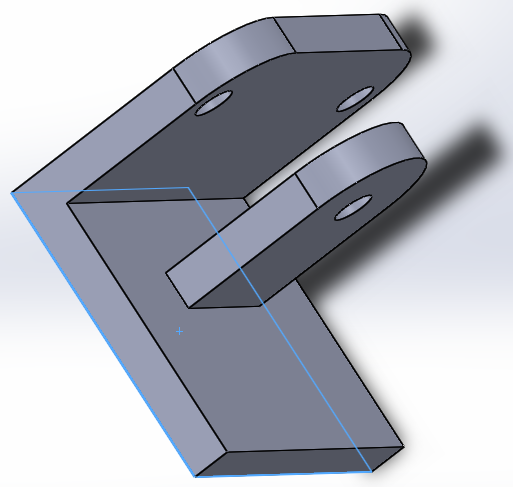
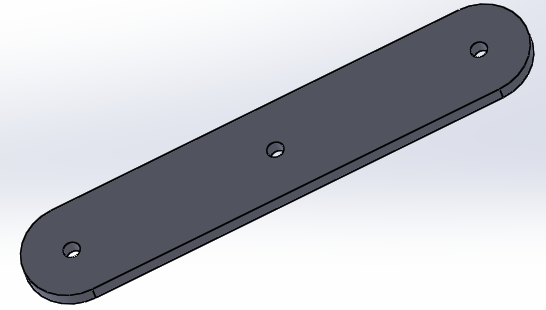
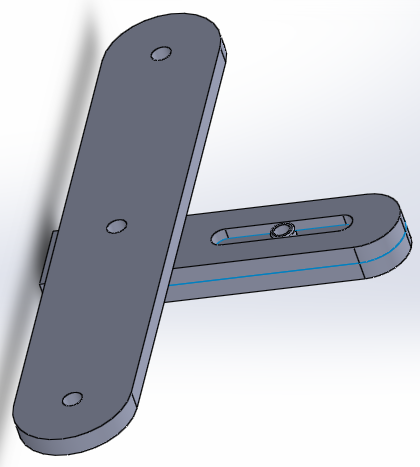
Dưới đây là thiết kế tổng quan cho bộ phận lái, bộ phận động cơ không thiết kế

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
* Network trên module 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: trả về client thông tin network của AP và STA của module wifi, và các SSID wifi mà module wifi bắt được
    - Byte 0: = ‘2’
  + Mode 3: gửi ssid và password yêu cầu module wifi kết nối tới
    - Gửi 2 gói
      * 30 <ssid>
      * 31 <pass>
  + Mode 4: gửi packet ‘4’: yêu cầu ngắt kết nối từ phía Server
  + Mode 5: gửi IP của AP và STA cho MCU từ module wifi đến MCU

## 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\_mode5()

Nhận IP của AP và STA của module wifi, lưu vào biến global, hiển thị lên LCD

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 hiển thị IP AP và STA của module wifi gửi đến, đã được lưu vào biến global

### 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()

Get IP AP và STA của Module Wifi, đồng thời gửi về cho Atmega16

Hàm được gọi khi phát hiện đã nhận được IP của AP khác (sau khi gọi mode 3 – yêu cầu kết nối đến 1 mạng ngoài)

### Phần mềm điều khiển trên máy tính

* 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

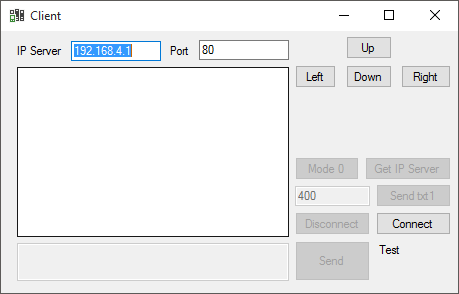
Hàm được gọi mỗi trước khi thực hiện một thao tác gửi nhận dữ liệu liên quan đến socket kết nối đến Server

### Phần mềm điều khiển trên di động

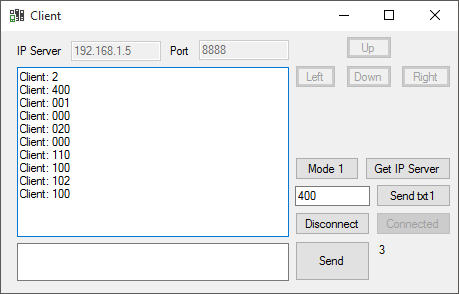
<Chưa tổng hợp>

# PHẦN 3: SẢN PHẨM KẾT QUẢ

## Giao diện phần mềm điều khiển trên máy tính

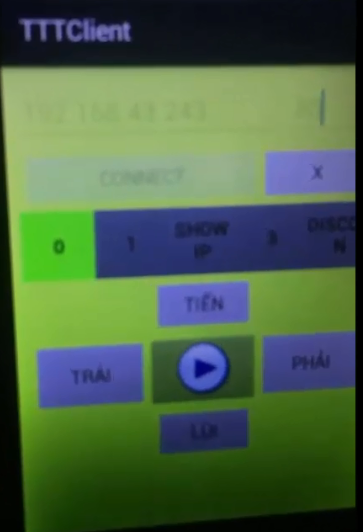


* Chọn IP Server và Port
* Click button Connect để tìm kiếm Server và kết nối
* Nếu có kết nối thì các button và text box sẽ được bật để sử dụng



* 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

## Giao diện phần mềm điều khiển trên di động



## Sản phẩm Xe

Video: <https://youtu.be/PCIph0KOMEc>

