NATIONAL INSTITUTE OF TECHNOLOGY

KARNATAKA, SURATHKAL

EE207: Electromagnetic Theory

Report On-

Wi-Fi Car

Professor: - ***Dr. Dastagiri Reddy***

An Effort by: -

Amar Pratap Singh (2110509)

Anurag C Narboli (2110512)

Bharat Singh (2110301)

DEPARTMENT OF

ELECTRICAL AND ELECTRONICS ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

SURATHKAL, MANGALORE – 575025

CONTENTS

1.Abstract

Automation, also known as automatic control, is the use of different control systems to operate any machinery or apparatus, such as a remote-controlled (toy) car, and to perform other valuable tasks with little to no human involvement. The design and implementation of a remote-controlled car using wi-fi technology on a computer or mobile device are proposed in this study. The ESP32S wireless module served as a transceiver (transmitter and receiver), the microcontroller, H-bridge L293D IC served as the motor controller, and the automobile was moved by four electric DC motors in order to complete this research project. Two objectives of this project are to expand the limitation range of a normal radio frequency car using wi-fi technology and also to create a ubiquitous technology for automobiles that operates in daily life with a control system. The test result shows that the controlled car can move in any direction

2.Introduction

The internet has grown to be a necessity in our life. Most people rely on the Internet to do numerous jobs and to ease their life. In actuality, the majority of individuals utilise a gadget that is online. The Internet serves a variety of purposes, including education, task-solving, process automation, entertainment, and data processing. The "Internet of Things," a new technology, has recently started to find a place in people's life to facilitate their varied activities without them feeling too exhausted. The Internet of Things (IoT), a new technological model envisioned as a global network of objects capable of talking with one another, can be divided into two categories: consumer IoT and industrial IoT. IoT is widely acknowledged as one of the most significant fields of the future.

The ability to remotely operate some electronic gadgets is one of the benefits of the Internet of Things. This is helpful in a variety of fields, including the military, agriculture, and more. An electronic vehicle (or robot) might be sent, for instance, to a difficult-to-reach location, such as beneath the wreckage of a home that has been severely damaged by an earthquake or other disaster. Or if we want to manage a military tool to observe the activities of the opposition. The ability to travel to places that a man cannot, does not want to, or is not advised is the main benefit of a remote-controlled car. The project's goal is to create a machine that can be operated remotely via a website. A Wi-Fi module will be utilised to connect to the internet, and four

DC motors will be used to manoeuvre the vehicle. Thus, the car can be transported to the four cardinal points using the website.

3.Scope

- It can be used as a monitoring and surveillance system.
- It can be used for locating the survivors hidden inside the debris.
- It can be used for parking assistance.
- This system can be used in defiance applications for detecting landmines in war field and for bomb detections by mounting a metal detector sensor on it.

4. Components Used

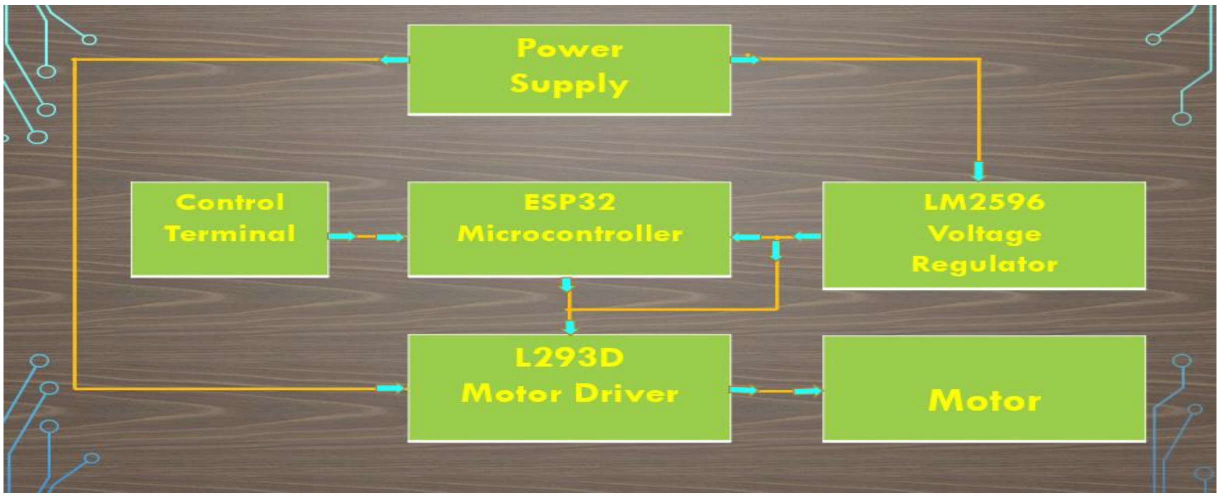
1- ESP32	4- Motors	7- Battery	10- Car Chassis
2- L293D	5- Breadboard	8- Switch	
3- LM2596	6- Wires	9- Multimeter	

5.Related Work

A mechatronic system is any mechanical-electrical device that uses a microprocessor to direct its motion. Due to the extraordinary rate of advancement in wireless communication, it is now possible to control a mechatronic system from practically everywhere on Earth where there are cellphones, personal computers, and Internet connections.

The MCU and Wi-Fi module serve as the foundation for the entire Wi-Fi car system, and is composed primarily of five components: the power supply, control terminal, Wi-Fi communication module, motor controller, and motor.

Figure below depicts the overall structure of the system.



As shown in above Figure, the control terminal of Wi-Fi car control can be a computer terminal, and it can also be a mobile phone.

Design of Wi-Fi intelligent car motor drive module

6. ESP32S

The ESP32 is a series of chip microcontrollers developed by Espressif.

It is very popular because of the following features: -

Low-cost: We can get an ESP32 starting at Rs 400-500, which makes it easily accessible to the general public;

Low-power: the ESP32 consumes very little power compared with other microcontrollers, and it supports low-power mode states like deep sleep to save power;

Wi-Fi capabilities: the ESP32 can easily connect to a Wi-Fi network to connect to the internet (station mode) or create its own Wi-Fi wireless network (access point mode) so other devices can connect to it—this is essential for IoT and Home Automation projects—we can have multiple devices communicating with each other using their Wi-Fi capabilities;

Bluetooth: the ESP32 supports Bluetooth classic and Bluetooth Low Energy (BLE)—which is useful for a wide variety of IoT applications;

Dual-core: most ESP32 are dual-core— they come with 2 Xtensa 32-bit LX6 microprocessors: core 0 and core 1.

Rich peripheral input/output interface—the ESP32 supports a wide variety of input (read data from the outside world) and output (to send commands/signals to the outside world) peripherals like capacitive touch, ADCs, DACs, UART, SPI, I2C, PWM, and much more.

Compatible with the Arduino "programming language": We can program the ESP32 in the same manner as the Arduino board.

ESP32 Specifications

Wireless connectivity Wi-Fi: 150.0 Mbps data rate with HT40

Processor: Tensilica Xtensa Dual-Core 32-bit LX6 microprocessor, running at 160 or 240 MHz

Memory:

- **ROM:** 448 KB (for booting and core functions)
- **SRAM:** 520 KB (for data and instructions)
- **RTC as SRAM:** 8 KB (for data storage and main CPU during RTC Boot from the deep-sleep mode)
- **RTC slow SRAM:** 8KB (for co-processor accessing during deep-sleep mode)

Low Power: ensures that we can still use ADC conversions, for example, during deep sleep

Peripheral Input/Output:

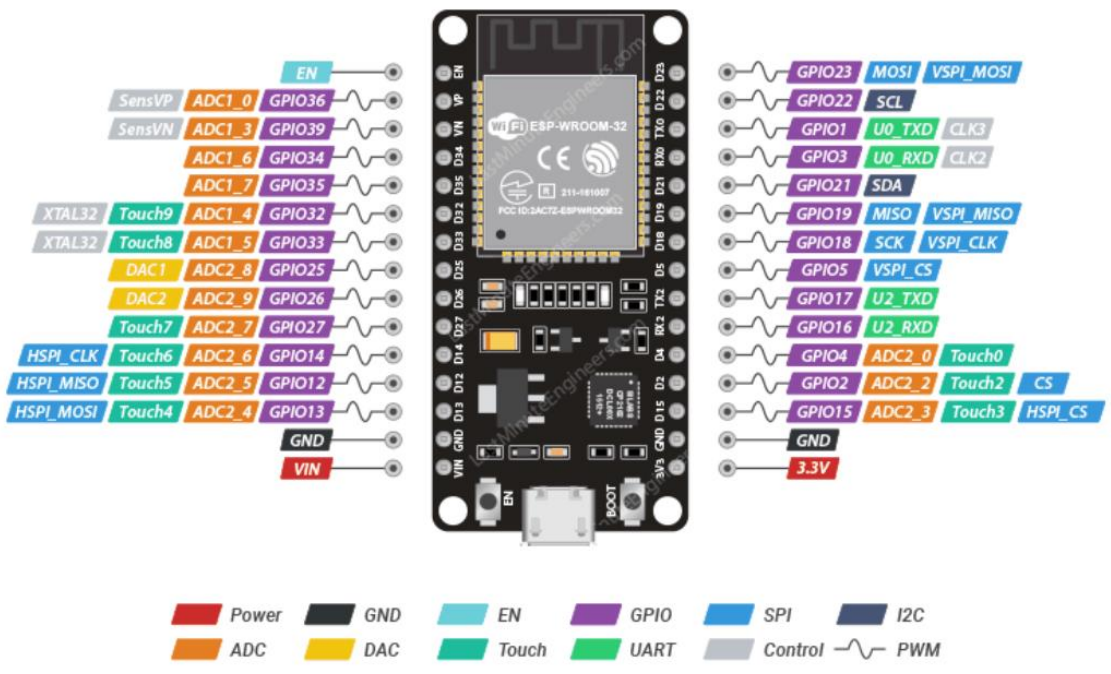
Peripheral interface with DMA that includes a capacitive touch

- ADCs (Analog-to-Digital Converter)
- DACs (Digital-to-Analog Converter)
- I²C (Inter-Integrated Circuit)
- UART (Universal Asynchronous Receiver/Transmitter)
- SPI (Serial Peripheral Interface)
- I²S (Integrated Inter Sound)
- RMII (Reduced Media-Independent Interface)
- PWM (Pulse-Width Modulation)

ESP32 Development Board

ESP-32 Pinouts

The following image shows the pinout of a 30-pin ESP32 Development Board.



Important ESP32 Peripherals

ESP32 Microcontroller has:

• 25 Programmable GPIOs
• 18 12-bit ADC Channels
• 2 8-bit DAC Channels
• 16 PWM Channels
• 3 UART Interfaces
• 10 Capacitive Touch Sensing GPIOs
• 16 RTC GPIOs

GPIOs (General Purpose Input/Output)

The most commonly used peripheral is the GPIO. ESP32 has 25 GPIO pins with each pin carrying out more than one function (only one will be active). In the software, users can set up a pin as a GPIO, an ADC, or a UART.

ADC and DAC pins are predefined and users have to use the manufacturer specified pins. But other functions like PWM, SPI, UART, I2C etc. can be assigned to any GPIO pin through program.

RTC GPIO

ESP32 has 16 RTC GPIOs, which are part of the RTC Low-Power subsystem. These pins can be used to wake ESP32 from deep sleep as external wake-up source.

ADC

ESP32 has two 12-bit SAR Analog to Digital Converter Modules with 8-channels and 10-channels each. So, ADC1 and ADC2 blocks combined together have 18 channels of 12-bit ADC.

With 12-bit resolution, the output Digital values will be in the range of 0 – 4093.

DAC

ESP32 Microcontroller has two independent 8-bit Digital to Analog Converter channels to convert digital values to analog voltage signals. The DAC has internal resistor network and uses power supply as input reference voltage.

The following two GPIO Pins are associated with DAC functionalities.

• DAC1 — GPIO25

• DAC2 — GPIO26

GPIO pins which are safe to use: -

GPIO	INPUT	OUTPUT	NOTES
0	Pulled up	**OK**	outputs PWM signal at boot, must be LOW to enter flashing mode
1	TX pin	**OK**	debug output at boot
2	OK	OK	connected to on-board LED, must be left floating or LOW to enter flashing mode
3	OK	RX pin	HIGH at boot
4	OK	OK	
5	OK	OK	outputs PWM signal at boot, strapping pin
6	X	X	connected to the integrated SPI flash
7	X	X	connected to the integrated SPI flash
8	X	X	connected to the integrated SPI flash

9	X	X	connected to the integrated SPI flash
10	X	X	connected to the integrated SPI flash
11	X	X	connected to the integrated SPI flash
12	OK	OK	boot fails if pulled high, strapping pin
13	OK	OK	
14	OK	OK	outputs PWM signal at boot
15	OK	OK	outputs PWM signal at boot, strapping pin
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	

Capacitive touch GPIOs

The ESP32 has 10 internal capacitive touch sensors. These can sense variations in anything that holds an electrical charge, like the human skin. So they can detect variations induced when touching the GPIOs with a finger. These pins can be easily integrated into capacitive pads and replace mechanical buttons.

PWM

The PWM Controller in ESP32 have 16 independent PWM waveform channels with configurable frequency and duty cycle. The PWM waveform can be used to drive motors and LEDs. The PWM signal's frequency, channel, GPIO pin, and UART can all be set in the application.

To set a PWM signal, we need to define these parameters in the code:

- Signal’s frequency;
- Duty cycle;
- PWM channel;
- GPIO where we want to output the signal.

Strapping Pins

The ESP32 chip has the following strapping pins:

- GPIO 0 (must be LOW to enter boot mode)
- GPIO 2 (must be floating or LOW during boot)
- GPIO 4
- GPIO 5 (must be HIGH during boot)
- GPIO 12 (must be LOW during boot)
- GPIO 15 (must be HIGH during boot)

These are used to put the ESP32 into bootloader or flashing mode.

Enable (EN)

Enable (EN) is the 3.3V regulator's enable pin. It's pulled up, so connect to ground to disable the 3.3V regulator.

ESP32 Control Digital Outputs

The GPIO we want to control must first be set as an OUTPUT. Use the pinMode() function as follows:

```
pinMode(GPIO, OUTPUT);
```

To control a digital output we just need to use the digitalWrite() function, that accepts as arguments, the GPIO (int number) we are referring to, and the state, either HIGH or LOW.

```
digitalWrite(GPIO, STATE);
```

All GPIOs can be used as outputs except GPIOs 6 to 11 (connected to the integrated SPI flash).

ESP32 Read Digital Inputs

First, set the GPIO we want to read as INPUT, using the pinMode() function as follows:

```
pinMode(GPIO, INPUT);
```

To read a digital input, like a button, we use the digitalRead() function, that accepts as argument, the GPIO (int number) we are referring to.

```
digitalRead(GPIO);
```

All ESP32 GPIOs can be used as inputs, except GPIOs 6 to 11 (connected to the integrated SPI flash).

analogRead() Function

Reading an analog input with the ESP32 using the Arduino IDE is as simple as using the analogRead() function. It accepts as argument, the GPIO we want to read:

```
analogRead(GPIO);
```

analogWrite()

To produce a PWM signal on a given pin we use the following function:

```
analogWrite(pin, value);
```

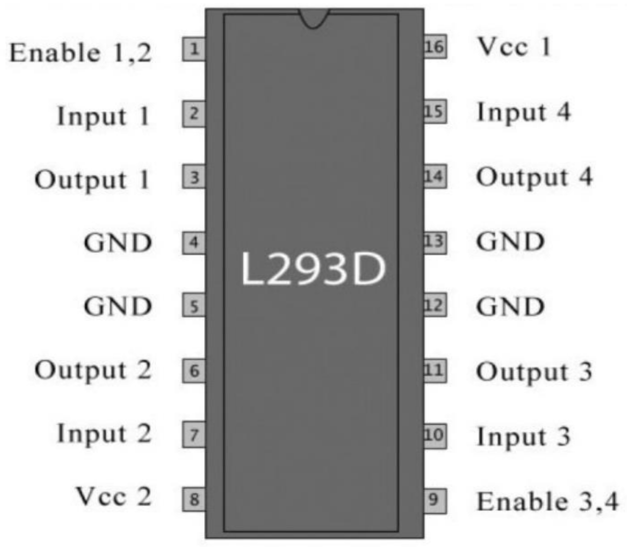
- **pin**: PWM may be used on pins 0 to 16
- **value**: should be in range from 0 to PWMRANGE, which is 255 by default. When value is 0, PWM is disable on that pin. A value of 255 corresponds to 100% duty cycle

7. L293D

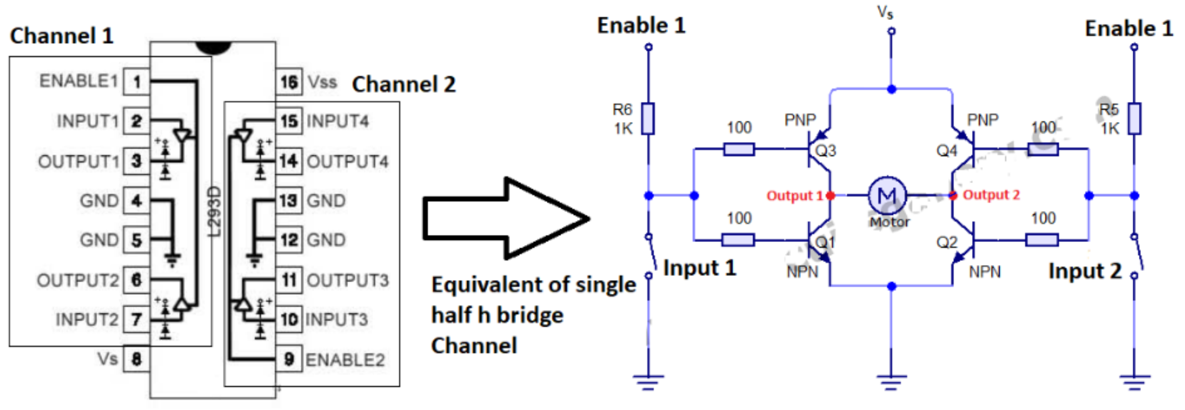
A motor driver is an integrated circuit chip that is usually used to control motors in autonomous robots. Motor driver act as an interface between ESP32 and the motors. The most commonly used motor driver ICs are from the L293 series such as L293D, L293NE, etc. These ICs are designed to control 2 DC motors simultaneously. L293D consists of two H-bridge. H-bridge is the simplest circuit for controlling a low current-rated motor. We will be referring to the motor driver IC as L293D only. L293D has 16 pins.

The L293D is a 16-pin IC, with eight pins on each side, dedicated to controlling a motor. There are 2 INPUT pins, 2 OUTPUT pins, and 1 ENABLE pin for each motor. L293D consists of two H-bridge. H-bridge is the simplest circuit for controlling a low current-rated motor.

L293D has two half-h bridge channels. Each channel can control a single motor independently. We can control four motors (2 motors in a same direction) with a single l293d motor driver. Each channel can control loads requiring 0 – 36 volts and 600 mA of current. Each channel has two input and two output pins. Our load(motor) is connected across the output pins of the channel. Controls are connected to input pins. Each channel is activated through an enable pin.

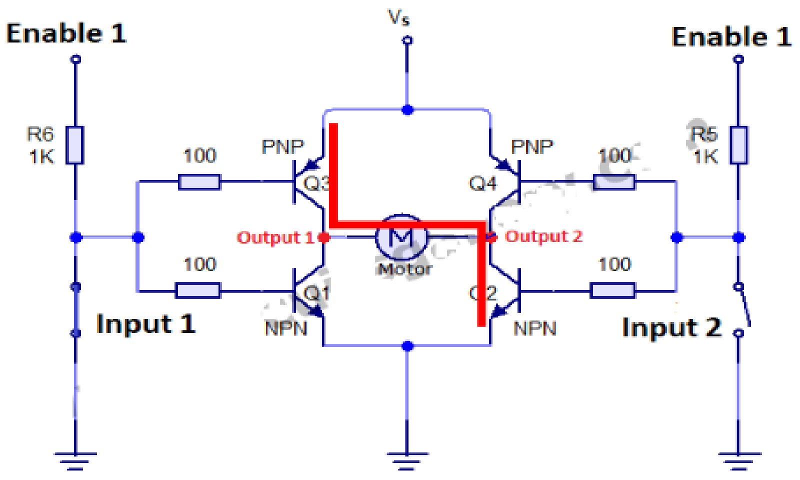


L293d Channels



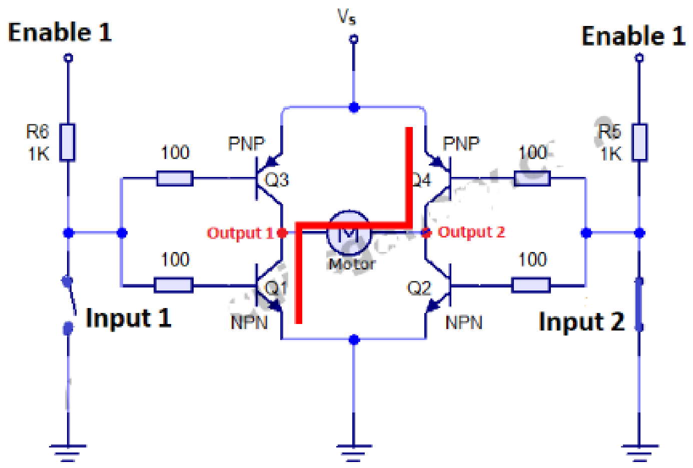
Working of L293D Channels

When input 1 is closed and input 2 is opened transistor Q3 and Q2 are opened/operational, allowing current to flow from them. The motor connected across the output 1 and 2 now starts rotating. The motor direction of rotation is clock wise. One PNP and one NPN transistor is activated when one input is closed and other is opened.



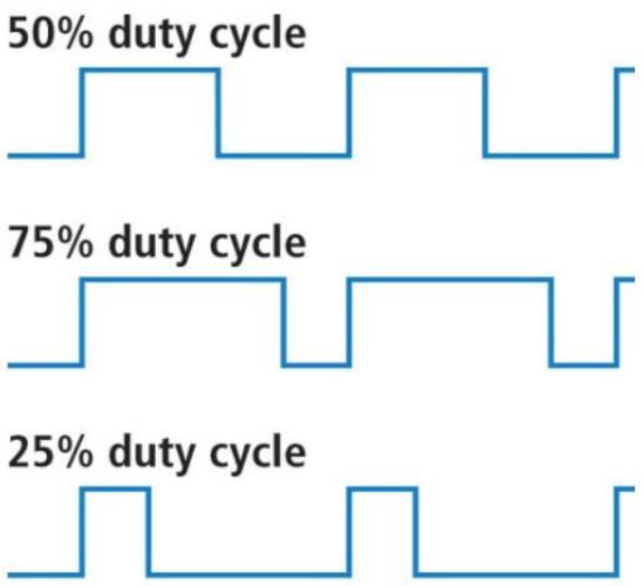
When input 1 is opened and input 2 is closed transistor Q4 and Q1 are operational. Current start flowing through this path and the motor starts moving in the anti-clock wise direction. Enable is still at high potential and vs is supplied battery required voltage.

If both the input 1 and input 2 are opened or closed to current path will be established and motor will not rotate. If any input is closed and enable is not at high potential than still the circuit will not work. Enable is a master pin and controls the whole circuit.

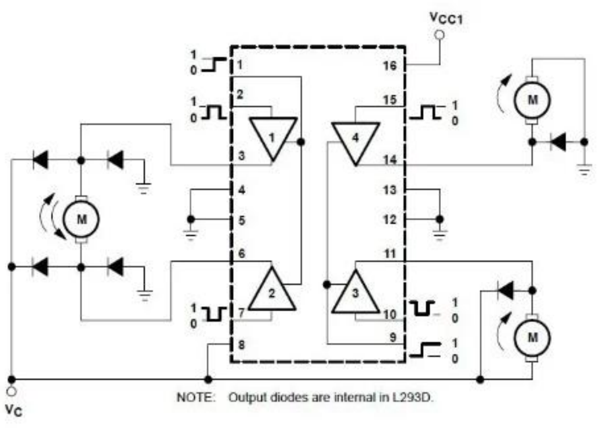


A PWM(pulse width modulated) signal can be applied to input pins to control the speed of motor. Enable pin can also be utilized to control the motor rotation speed. By enabling and disabling the enable pin, individual channels can be enable/disable rapidly producing a pulse width modulated signal which alternatively lowers or increases the rotation speed of dc motor.

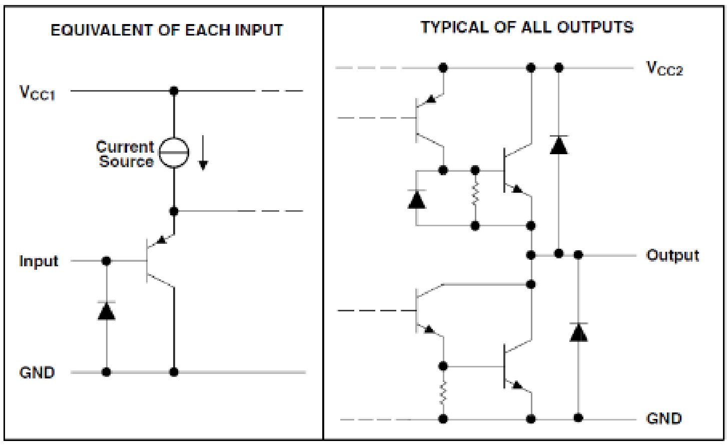
In a pulse width modulation signal output of duty cycle of the signal is controlled. The duty cycle is the amount of time the signal is high in a given period of a digital signal. Duty cycles are shown diagrammatically below. In the first wave duty cycle is 50 %. It means that the signal is high for 50 % of the period. For example, the period is 2 seconds. With a 50 % duty cycle, the output remains high for 1 second and low for 1 second. If the duty cycle is 75 %. Output remains high for 1.5 seconds and low for 0.5 seconds.



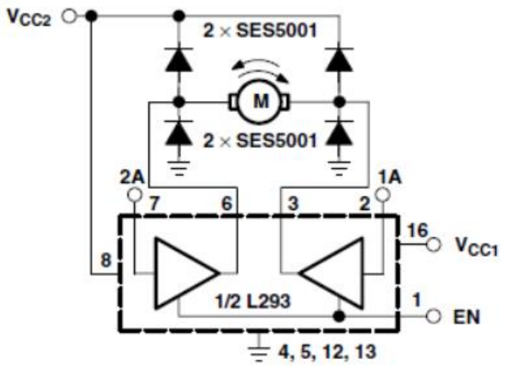
L293D Internal structure



The equivalent of input circuit and output circuit is shown below. Input side is simple just a transistor with a protection diode. At output side we have a Darlington circuit with high current gain and protection diodes. The rotation of motors connected at the output of l293d and the input output truth table is shown below.



equivalent circuit

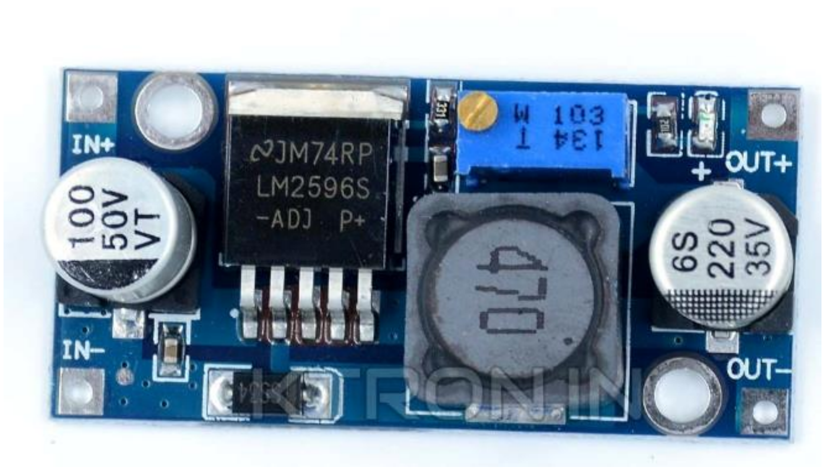


EN	1A	2A	FUNCTION
H	L	H	Turn right
H	H	L	Turn left
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Fast motor stop

L = low, H = high, X = don't care

8. LM2596

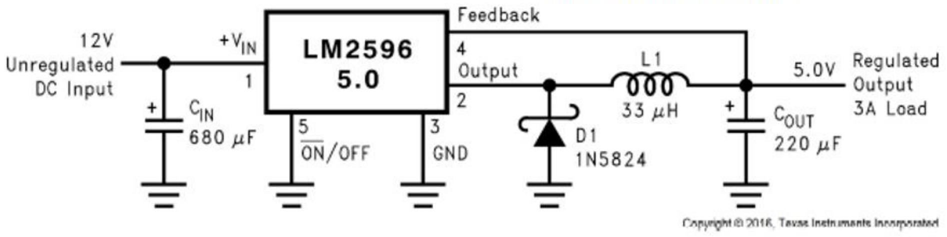
The LM2596 series of regulators are monolithic integrated circuits that provide all the active functions for a step-down (buck) switching regulator (DC-DC converter), capable of driving a 3-A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3 V, 5 V, 12 V, and an adjustable output version. Requiring a minimum number of external components, these regulators are simple to use and include internal frequency compensation, and a fixed frequency oscillator. The LM2596 series operates at a switching frequency of 150 kHz, thus allowing smaller sized filter components than what would be required with lower frequency switching regulators. Available in a standard 5-pin TO-220 package with several different lead bend options, and a 5-pin TO-263 surface mount package.



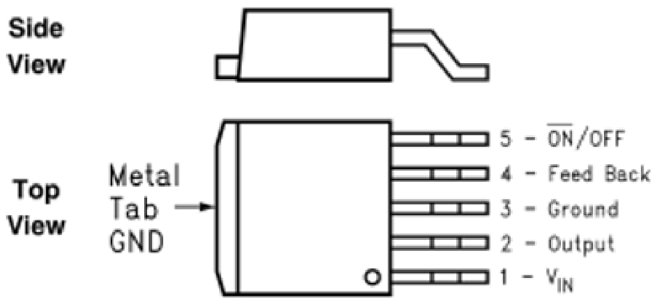
Features

- 3.3-V, 5-V, 12-V, and adjustable output versions
- Adjustable version output voltage range: 1.2-V to 37-V ±4% maximum over line and load conditions
- Available in TO-220 and TO-263 packages
- 3-A output load current
- Input voltage range up to 40 V
- Requires only four external components
- Excellent line and load regulation specifications
- 150-kHz Fixed-frequency internal oscillator
- Uses readily available standard inductors

(1) For all available packages, see the orderable addendum at the end of the data sheet.



Pin Configurations and Functions



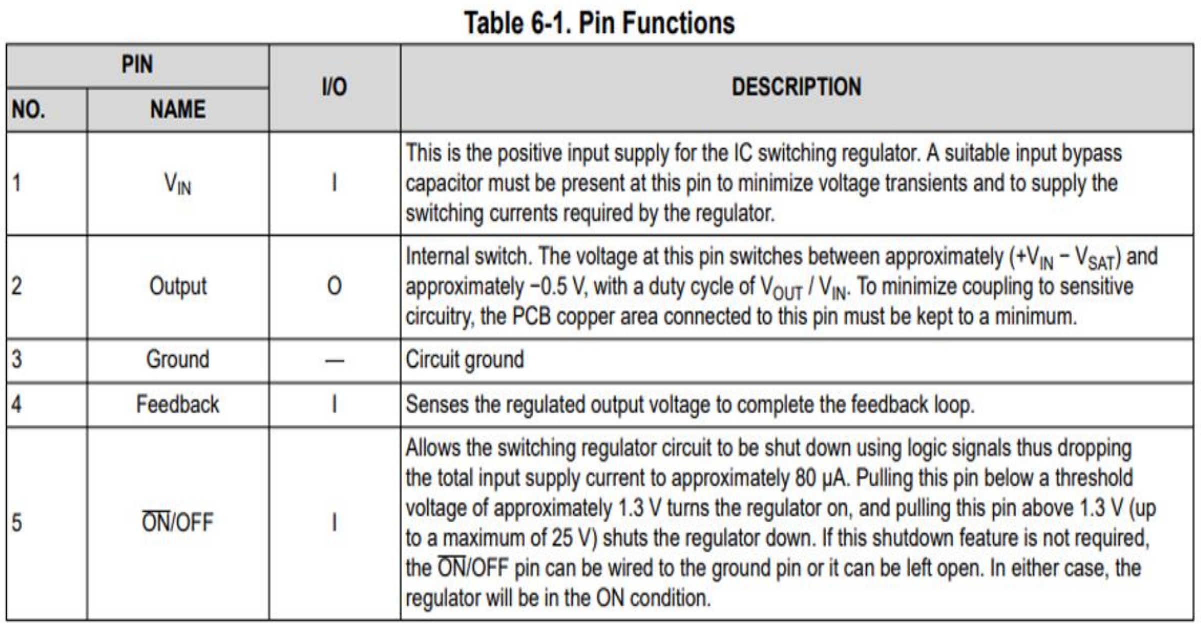


Table 6-1. Pin Functions

PIN		I/O	DESCRIPTION
NO.	NAME		
1	V_{IN}	I	This is the positive input supply for the IC switching regulator. A suitable input bypass capacitor must be present at this pin to minimize voltage transients and to supply the switching currents required by the regulator.
2	Output	O	Internal switch. The voltage at this pin switches between approximately $(+V_{IN} - V_{SAT})$ and approximately −0.5 V, with a duty cycle of V_{OUT} / V_{IN}. To minimize coupling to sensitive circuitry, the PCB copper area connected to this pin must be kept to a minimum.
3	Ground	—	Circuit ground
4	Feedback	I	Senses the regulated output voltage to complete the feedback loop.
5	$\overline{\text{ON}}$/OFF	I	Allows the switching regulator circuit to be shut down using logic signals thus dropping the total input supply current to approximately 80 µA. Pulling this pin below a threshold voltage of approximately 1.3 V turns the regulator on, and pulling this pin above 1.3 V (up to a maximum of 25 V) shuts the regulator down. If this shutdown feature is not required, the $\overline{\text{ON}}$/OFF pin can be wired to the ground pin or it can be left open. In either case, the regulator will be in the ON condition.

8.1 Input Capacitor (CIN)

This capacitor prevents large voltage transients from occuring at the input, and provides the instantaneous current required each time the switch turns ON.

8.2 Output Capacitor (COUT)

An output capacitor is required to filter the output and provide regulator loop stability.

8.3 Catch Diode

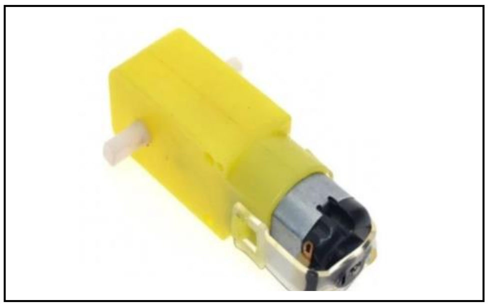
Buck regulators require a diode to provide a return path for the inductor current when the switch turns off. This must be a fast diode and must be placed close to the LM2596 using short leads and short printed-circuit traces. Because of their very fast switching speed and low forward voltage drop, Schottky diodes provide the best performance, especially in low output voltage applications (5 V and lower).

9. BO Motors

BO (Battery Operated) light weight DC geared motor which gives good torque and rpm at lower voltages. This motor can run at approximately 150 RPM when driven by a single Li-Ion cell. Great for battery operated light weight robots. A specific type of DC geared motors that can be operated through battery and that why known as Battery Operated (BO) motors. It is used for light weight applications mostly. Available in different torque and RPM

Features:

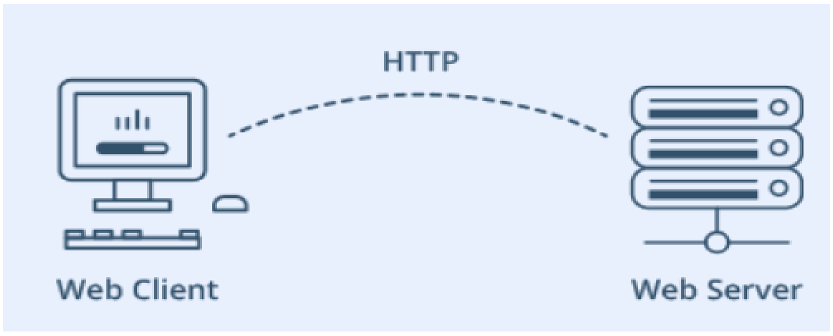
- Input Voltage(V): 4.5 - 12V
- Current rating: 0.07A (maximum on load)
- Speed(RPM): 200 RPM+-10%



Wi-Fi communications

Wi-Fi networks transmit information over the air using radio waves, which are a type of electromagnetic radiation with wavelengths in the electromagnetic spectrum longer than infrared light. Wi-Fi radio waves typically have the frequency of either 2.4 GHz or 5 GHz. The typical range of a standard Wi-Fi network can reach up to 100 meters in the open air. Buildings and other materials reflect the signal however, making most Wi-Fi networks far narrower than that. The strength of the antenna and the frequency broadcast can also impact the effective range of the network. Higher frequencies like 5 GHz and 60 GHz have far shorter effective ranges than 2.4 GHz.

A web server is a place where web pages are stored, processed, and served to web clients. A web client is just a web browser that we use on our computers and phones. A web client and a web server communicate using a special protocol known as Hypertext Transfer Protocol (HTTP).



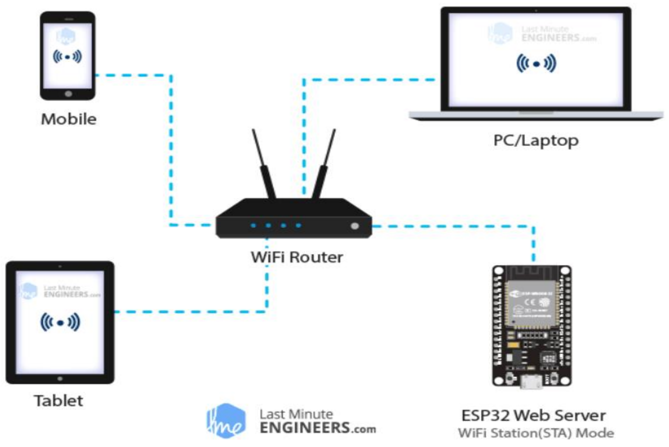
In this protocol, a client starts a conversation by sending an HTTP request for a specific web page. The server then sends back the content of that web page or an error message if it can't find it.

ESP32 Operating Modes

ESP32 can operate in three modes: Station (STA) mode, Soft Access Point (AP) mode, and both simultaneously.

Station (STA) Mode

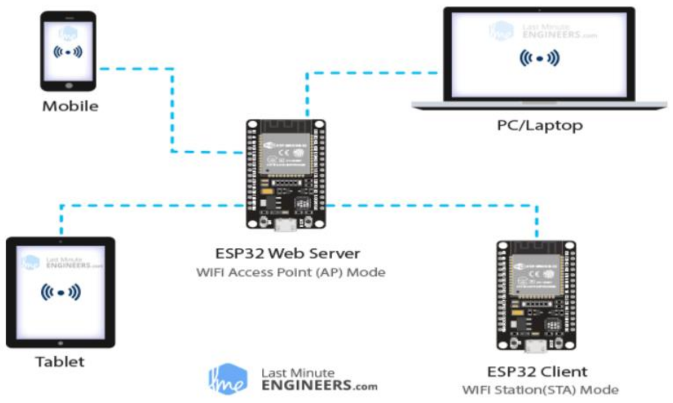
In Station (STA) mode, the ESP32 connects to an existing Wi-Fi network (the one created by wireless router).



In STA mode, the ESP32 obtains an IP address from the wireless router to which it is connected. With this IP address, it can set up a web server and serve web pages to all connected devices on the existing Wi-Fi network.

Soft Access Point (AP) Mode

In Access Point (AP) mode, the ESP32 sets up its own Wi-Fi network and acts as a hub, just like a Wi-Fi router. However, unlike a Wi-Fi router, it does not have an interface to a wired network. So, this mode of operation is called Soft Access Point (soft-AP).



In AP mode, the ESP32 creates a new Wi-Fi network and assigns it an SSID (the network's name) and an IP address. With this IP address, it can serve web pages to all connected devices.

10.Software Implementation

10.1 Arduino IDE

A program for Arduino may be written in any programming language for a compiler that produces binary machine code for the target processor. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio. IDE - Integrated Development Environment is a software application that combines all of the features and tools needed by a software developer. IDEs increase programmer productivity by combining common activities of writing software into a single application: editing source code, building executables, and debugging. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple on click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

10.2 ESP32 Libraries

Including the following libraries is a must to use the IoT functions and features of ESP32

1- WiFi
2- ESPAsyncWebServer

3- analogWrite
4- ESPAsyncTCP ("ESPAsyncWebServer" library depends on "ESP32AsyncTCP" so "ESP32AsyncTCP" library also need to be installed.)

10.3 Arduino Code

```
//Including the following libraries is a must to use the IoT functions and features of ESP32

#include "Wi-Fi.h"

#include "ESPAsyncWebServer.h"

#include "analogWrite.h"

const char* ssid = "Wi-Fi Car"; // This will show up when we turn on wer mobile WI-FI

const char* password = "EMTProject";  // change to more unique password


AsyncWebServer server(80); // These will start the webserver // If we don't know much, Ignore this line


// These pins are the Enable pins of the L293D motor driver which connects to ESP32 GPIO pins to implement the PWM function

int enable1_2 = 21; // choose the GPIO pin of ESP32;

int enable3_4 = 22; // choose the GPIO pin of ESP32;


// These pins are the input pins of l293d on the left side

int inp1 = 16; // Choose wer GPIO pin of ESP32 for the input 1

int inp2 = 17; // Choose wer GPIO pin of ESP32 for the input 2


// These pins are the input pins of l293d on the right side

int inp3 = 18; // Choose wer GPIO pin of ESP32 for the input 3

int inp4 = 19; // Choose wer GPIO pin of ESP32 for the input 4


int led = 2; // until now we must know what is the inbuilt led pin number of ESP32.

void setup(){
 // Fill in the blanks


 pinMode(enable1_2, OUTPUT);
 pinMode(enable3_4, OUTPUT);


 // The inputs
 pinMode(inp1, OUTPUT);
 pinMode(inp2, OUTPUT);
 pinMode(inp3, OUTPUT);
 pinMode(inp4, OUTPUT);


 pinMode(led, OUTPUT);


// We use the following function to run the bot at variable speed.
 analogWrite(enable1_2, 255); // analog write "255" corresponds to digital write "1"
 analogWrite(enable3_4, 255);


 Wi-Fi.softAP(ssid, password);  // This sets wer ESP32's name as per above mentioned


 IPAddress IP = Wi-Fi.softAPIP();


// A bit of WEB DEV stuff
 server.on("/control", HTTP_GET, [](AsyncWebServerRequest *request){


   if(request->hasArg("up")){
     // We have to develop the logic that, when the Button "up" on the webpage is pressed, Then the
ESP32 executes the following commands
     // Refer to the this for the webpage photo
     digitalWrite(inp1, HIGH);
     digitalWrite(inp2, LOW);
```

```
  digitalWrite(inp3, HIGH);
  digitalWrite(inp4, LOW);

  digitalWrite(led, HIGH);
}

if(request->hasArg("down")){
  digitalWrite(inp1, LOW);
  digitalWrite(inp2, HIGH);
  digitalWrite(inp3, LOW);
  digitalWrite(inp4, HIGH);

  digitalWrite(led, HIGH);
}

if(request->hasArg("left")){
  digitalWrite(inp1, HIGH);
  digitalWrite(inp2, LOW);
  digitalWrite(inp3, LOW);
  digitalWrite(inp4, HIGH);

  digitalWrite(led, HIGH);
}
if(request->hasArg("right")){
  digitalWrite(inp1, LOW);
  digitalWrite(inp2, HIGH);
  digitalWrite(inp3, HIGH);
  digitalWrite(inp4, LOW);

  digitalWrite(led, HIGH);
}
```

```
    if(request->hasArg("stop")){
// if '255' is the equivalent to digital '1', and '0' is eqvivalent to digial '0', We vary the pwm values to vary the speed of the motor
      analogWrite(enable1_2, 0);
      analogWrite(enable3_4, 0);
      digitalWrite(inp1, LOW);
      digitalWrite(inp2, LOW);
      digitalWrite(inp3, LOW);
      digitalWrite(inp4, LOW);

      digitalWrite(led, LOW);
    }

    if(request->hasArg("start")){
      analogWrite(enable1_2, 255 );
      analogWrite(enable3_4, 255);

      digitalWrite(led, HIGH);
    }

    request->send_P(200, "text/plain", "");
  });
  server.begin();
}
void loop(){
}
```

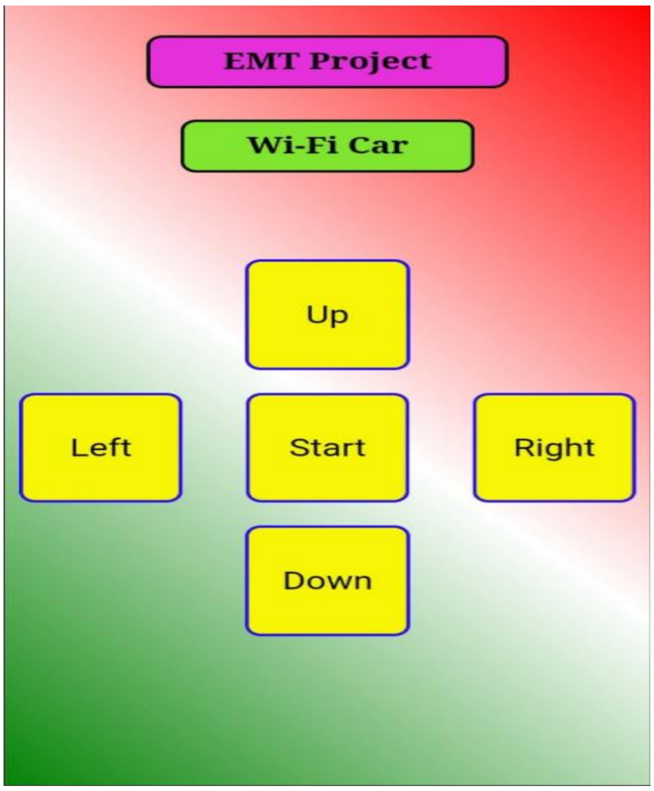
10.4 Web Interface

```
<html>

<head>

  <meta name="viewport" content="width=device-width">

</head>

<style>


 #title{

  margin: 26px 71px;

  padding: 9px;

  font-size: 19px;

  border: 2px solid rgb(5, 5, 5);

  border-radius: 9px;

  background-color: rgb(229 48 219);


 }

 #Wi-Fi{

  margin: 15px 90px;

  padding: 9px;

  font-size: 19px;

  border: 2px solid rgb(5, 5, 5);

  border-radius: 9px;

  background-color: rgb(130 229 48);


 }


 .unselectable {

  -webkit-touch-callout: none;

  -webkit-user-select: none;

  -khtml-user-select: none;
```

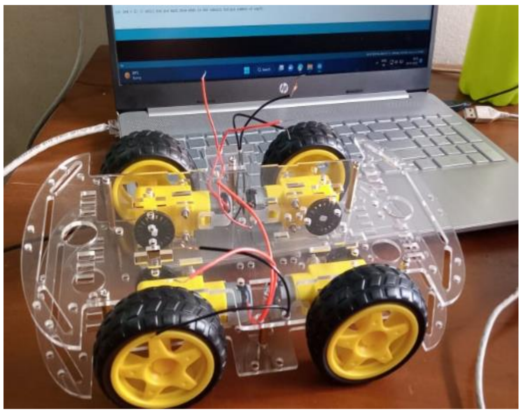
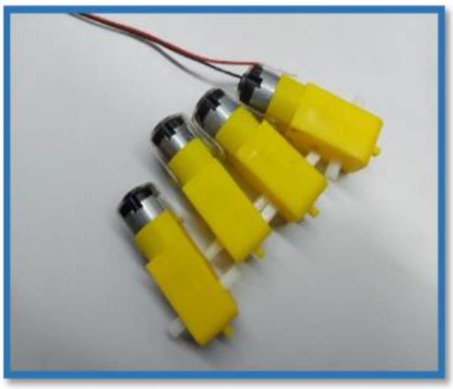
```
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;

    border: 2px solid rgb(31, 6, 252);
    border-radius: 9px;
    background-color: rgb(249 245 7);
    font-size: 20px;
}
.box:hover{
  color: rgb(255, 102, 0);
  background-color: rgb(0, 247, 255);
}



</style>

<body style="background-image: linear-gradient(45deg,green,white,red);">

<div style="text-align:center;">
  <h1 id="title">EMT Project</h1>
</div>

<div style="text-align:center;">
  <h1 id="Wi-Fi">Wi-Fi Car</h1>
</div></br></br></br>

<div style="text-align:center;">
        <button class="unselectable box" ontouchstart="up()" ontouchend="off()"
style="height:92px;width:92px;"> Up </button>
</div>
```

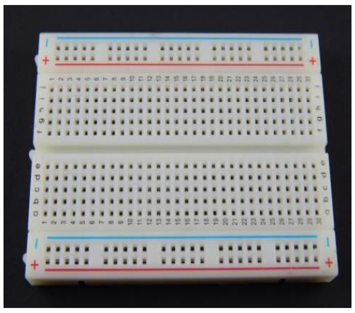
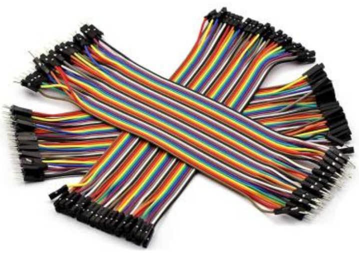
```
</br>

  <div style="float:left;">
   <button class="unselectable box" ontouchstart="left()" ontouchend="off()" style="height:91px;width:91px;"> Left </button>
  </div>

  <div style="float:right;">
   <button class="unselectable box" ontouchstart="right()" ontouchend="off()" style="height:91px;width:91px;"> Right </button>
  </div>

  <div style="text-align:center;">
            <button class="unselectable box" id="toggle" ontouchstart="toggle()" style="height:91px;width:91px;">Start
</button>
  </div>
  </br>

<div style="text-align:center;">
      <button class="unselectable box" ontouchstart="down()" ontouchend="off()" style="height:92px;width:92px;"> Down </button>                    </div>

<script>
   var start = 1;
   var url = "http://192.168.4.1/control?";
   function up(){
      fetch(url+"up")
   }

   function down(){
      fetch(url+"down")
   }

   function left(){
      fetch(url+"left")
   }

   function right(){
      fetch(url+"right")
   }

   function off(){
      fetch(url+"off")
   }

   function toggle(){
      if(start==1){
             start=0;
        document.getElementById("toggle").innerHTML="Start"
             fetch(url+"stop");
          }else{
        start=1;                                    document.getElementById("toggle").innerHTML="Stop"
        fetch(url+"start");
          }
   }
</script>
</body>
</html>
```

Controlling Website

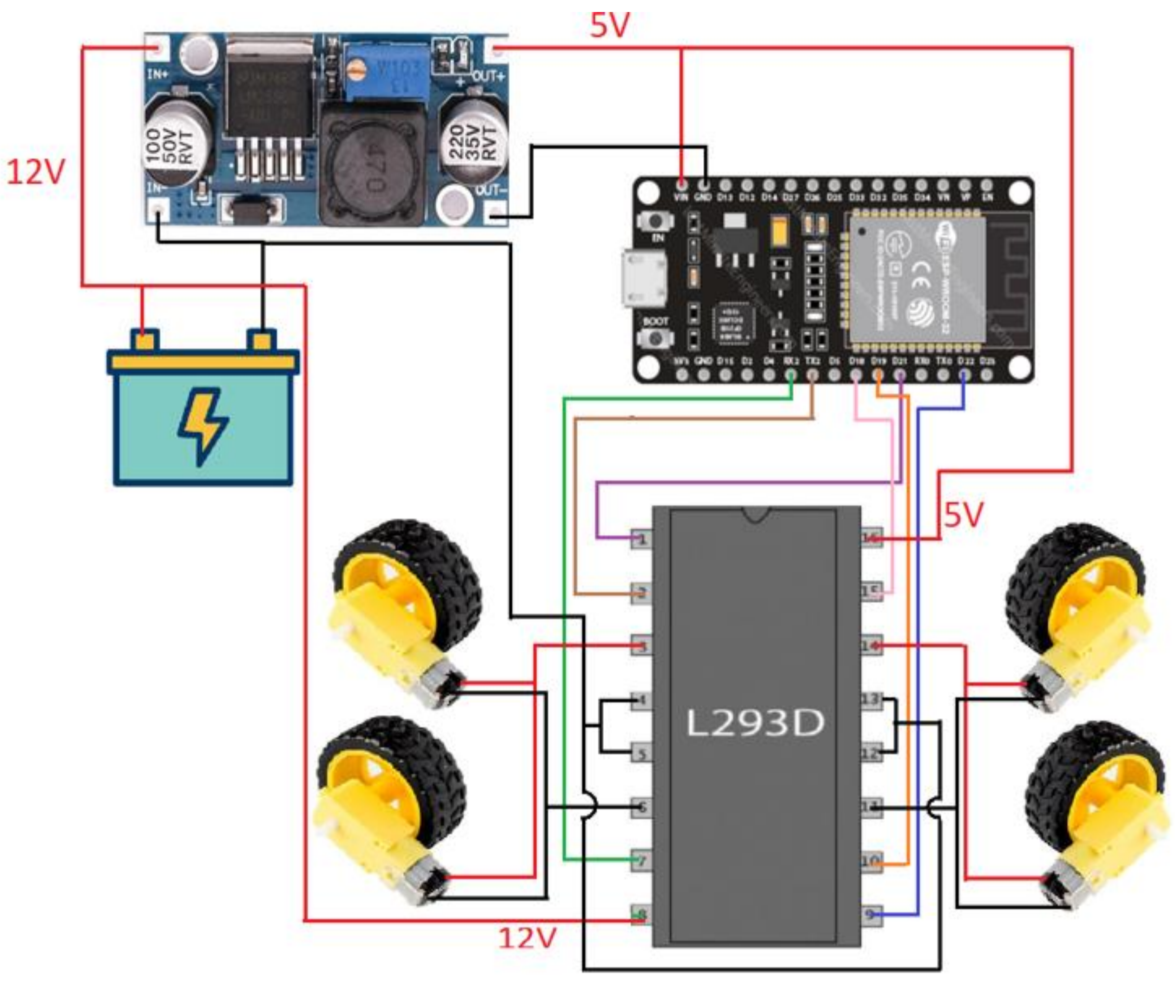


Materials Used

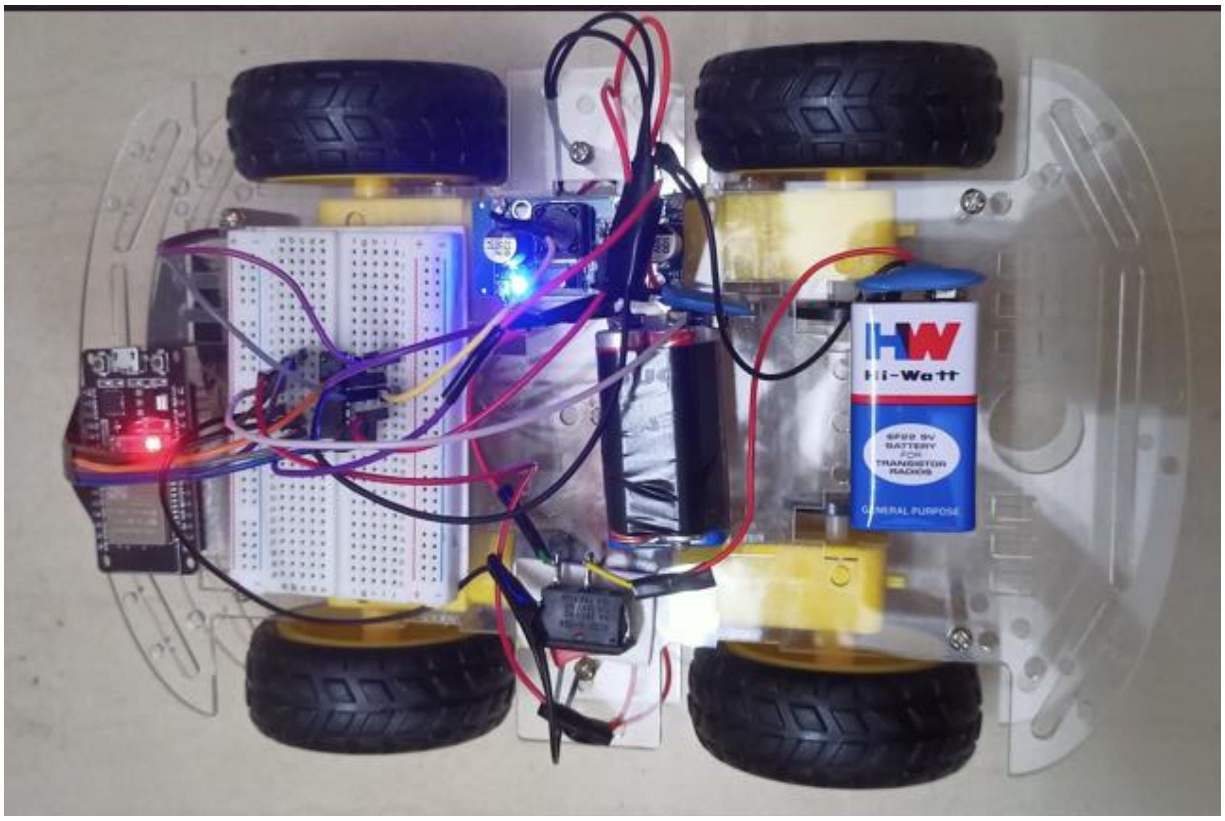



Connection diagram



Model



11. Conclusion

The goal of this project was to design and build a Wi-Fi remote control car using a mobile device. We used our own prior knowledge of programming and took help from our seniors, friends, and google to write the code. A simple Wi-Fi control car is implemented in this project where a car is controlled using a web page over Wi-Fi Network. In this project, we use Arduino which is an open-source device that has been the brain for numerous projects. We implement code for Wi-Fi car in ESP 32 with the help of Arduino.

We can also make this project with advanced features like integrating a camera and accessing the feed live on the browser.

12.Learning outcomes

When discussing the learning outcomes related to the group project they can be discussed under many different areas. The main focus of this project was on making a wi-fi car with the help of an ESP-32 microcontroller. This helped us a lot in getting proper insight into this module. In this project, we got an opportunity to understand how to apply theoretical knowledge practically in real-world situations. This group project helped us a lot in developing our skills in working as a team and also contributed to developing our research skills.

13. References

- https://www.engineersgarage.com/l293d-pin-description-and-working/
- https://randomnerdtutorials.com/getting-started-with-ESP32/
- https://lastminuteengineers.com/ESP32-pinout-reference/
- https://www.electronicshub.org/ESP32-pinout/
- https://www.ijert.org/remote-monitoring-system-based-on-a-wi-fi-controlled-car-using-arduino
- https://www.ijsr.net/archive/v8i5/ART20197508.pdf
- https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/
- https://www.digikey.in/en/products/detail/texas-instruments/L293N/1509928
- https://www.instructables.com/How-to-Use-DC-to-DC-Buck-Converter-LM2596/
- LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator datasheet (Rev. F) (ti.com)
- https://en.wikipedia.org/wiki/ESP32
- https://www.studiopieters.nl/ESP32-pinout/
- https://www.arduinolibraries.info/libraries/ESP32-analog-write
- https://sproboticworks.com/shop/products/plastic-gear-motor-with-pcb.html