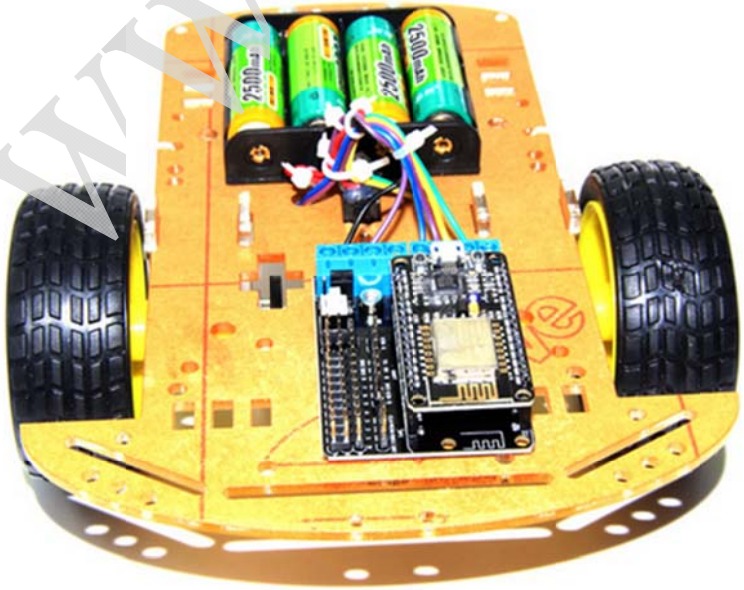
Doctors of Intelligence & Technology (DOIT)

User Manual for 2WD Car Chassis



11 Jan. 2016

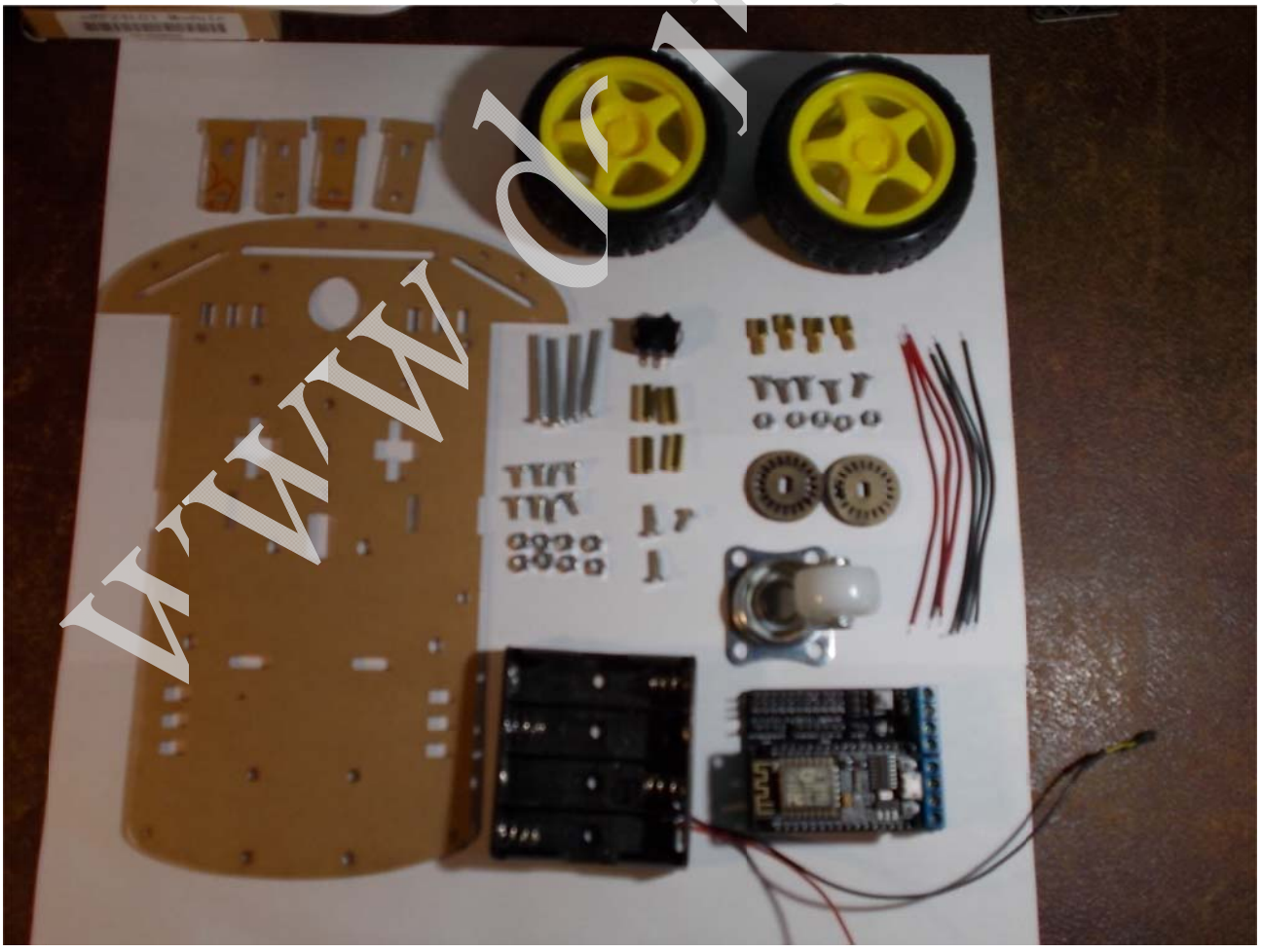
User Manual for 2WD Car Chassis

Note:
Pictures of the completed car can be found here http://www.smartarduino.com/view.php?id=94572
And there is an " Instructable " at http://www.instructables.com/id/A-very-cheap-ESP8266-WiFi-smart-car-controlled-by-/?ALLSTEPS

There are some instructions for the chassis Kit, which I think is the same as their Arduino kit and are included within the these few words.

Hardware building the car.

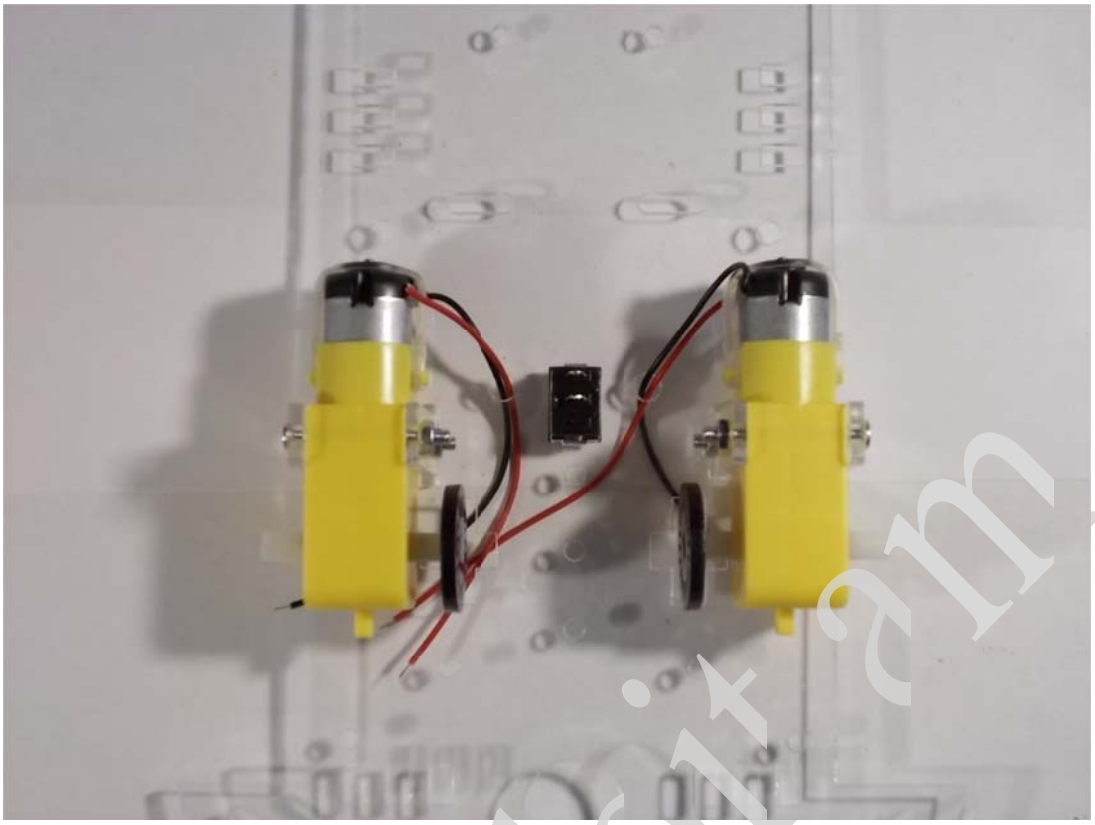
Check Items in the kit are they all there?



Remove paper from Plastic components, Chassis, Motor mounts and rotation encoders.

The chassis instructions now say to fit the motors, but I think that it would be better to solder on the motor wires before fitting the motors.

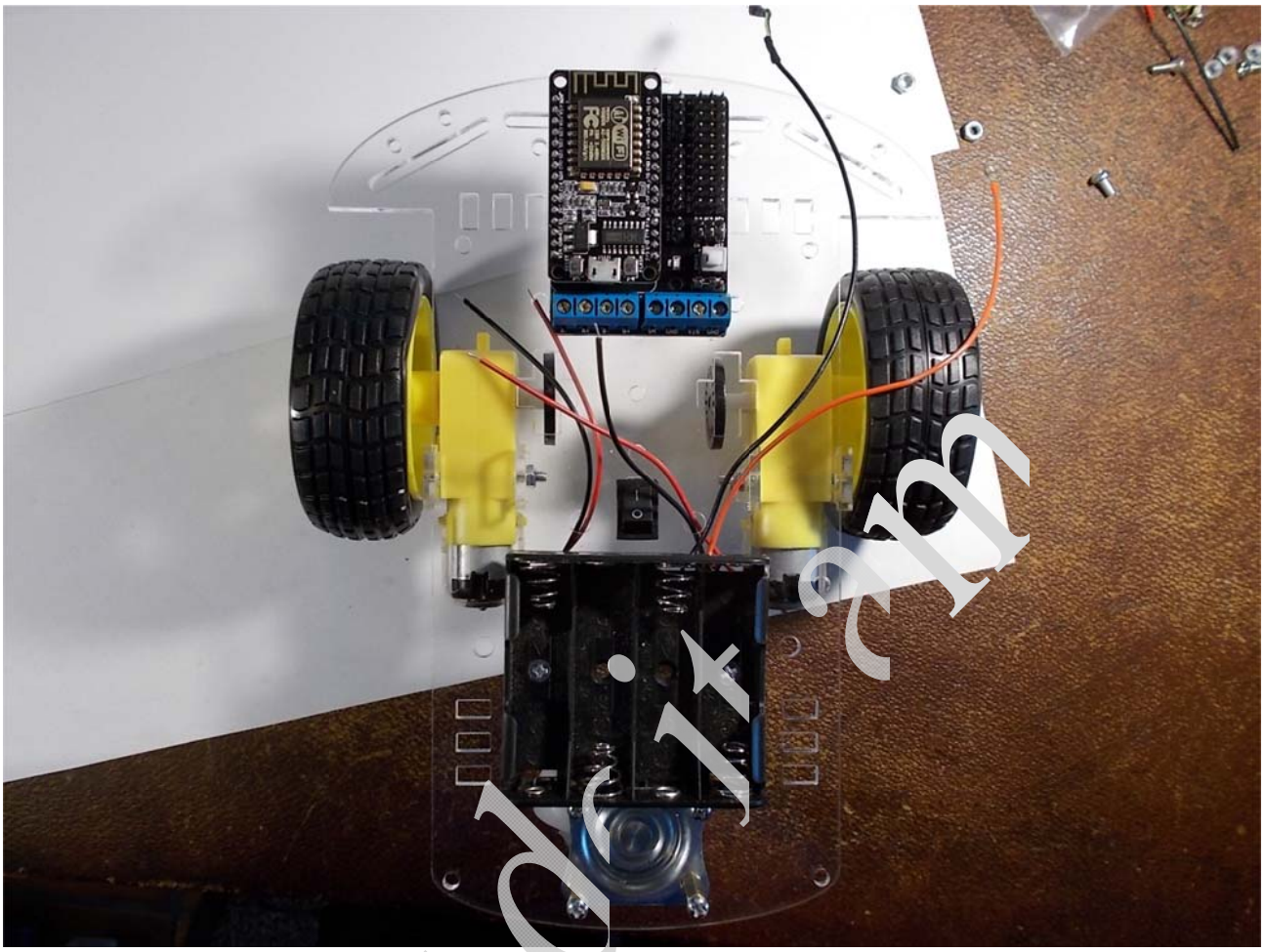
Now the motors and power switch can be fitted.



The battery box is fitted next using the large slots to allow alignment.

The four supports for the castor wheel can now be fitted, only do the screws up finger tight to allow some adjustment when fitting the caster wheel. Note that the base of the caster wheel is not square but rectangular, so if the holes do not align up then rotate 90 degrees and things should be a lot better. The caster wheel can now be fitted to the spacers , again only do the screws up finger tight until all screws are fitted. Now all screws can be tightened and the caster wheel and supports should fully align.

I had in a separate bag, the wires, four brass standoffs and some nuts and bolts. I originally thought that these were for fixing the Motor Board and ESP8266-12E, but I could not find any suitable holes in the chassis. I suppose that holes could be drilled for the standoffs in the chassis, but I prefer my old standby of double sided sticky tape. The variant that I use has a thin layer of foam that is sticky on both sides, sold for fixing things like door bell switched to plastic door frames. You can make your own if you have the scotch type of double sided tape, just the thin plastic type, by using the foam packing material that protected the SmartCar and building up a sandwich of tape and foam.



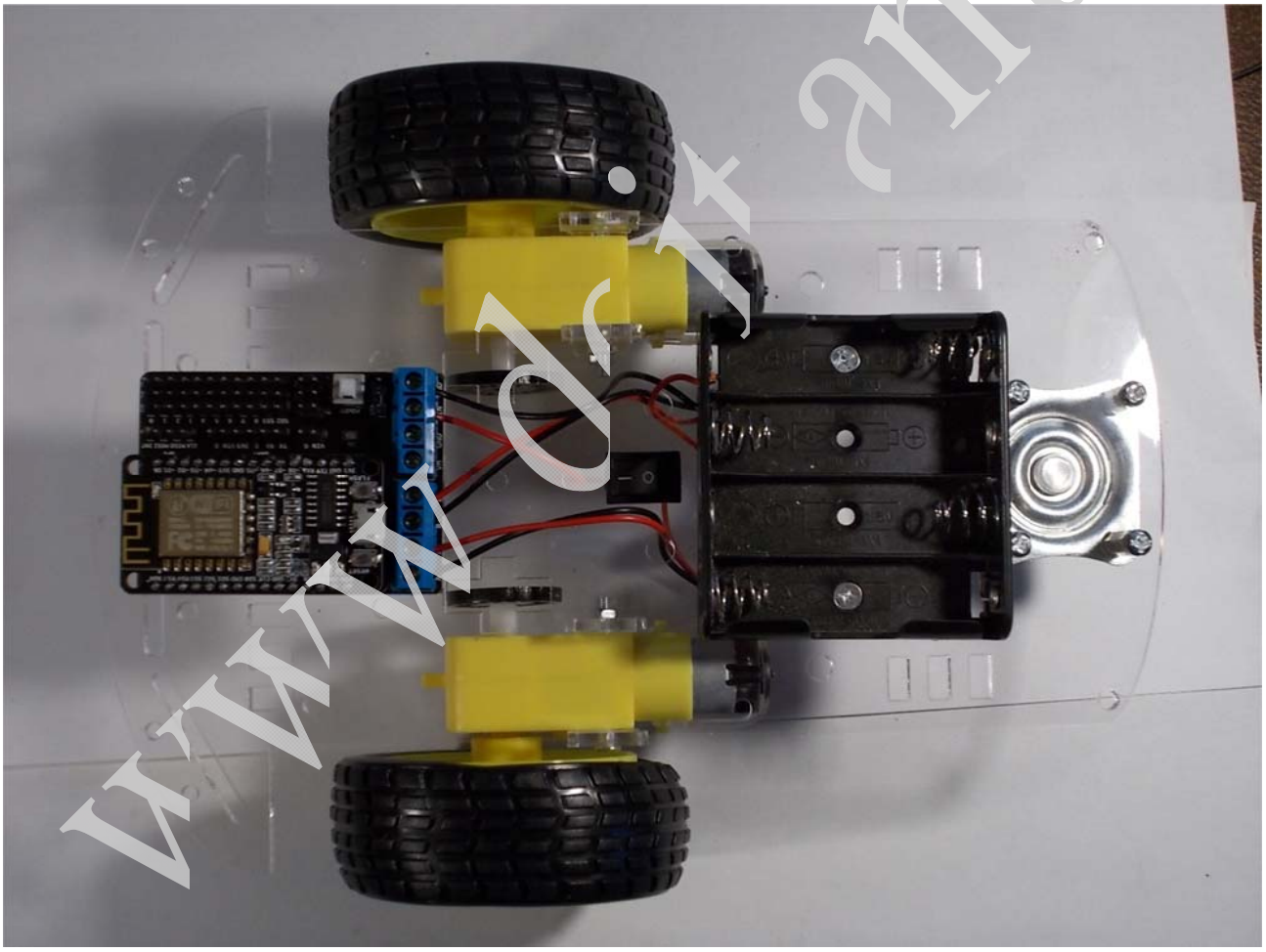
The connections to the motor board are via the eight blue screw connectors at one end. Looking at them you will see that they are labeled: A-, A+, B-, B+, VM, GND, VIN and GND. The board is to be placed at the front of the car, so I suggest that motor connection A- and A+ are used for the left hand motor and B- and B+ are used for the right hand motor. I found that the right hand motor had the shortest lead so I connected that up before fixing the motor board, to make sure that it would reach. At this stage it does not matter which motor wire goes into which connector, but just for convention I put the red wire into the + connector and the black wire into the – connector. When the time comes to run the car the wires can be swapped if the motors are not running in the correct direction.

The power switch now needs to be connected. Take the positive wire from the battery holder and pass it down through the chassis, I used the same hole as the wires for the right hand motor, and solder it to one of the switch lugs. Take the remaining red wire and solder it to other switch lug and take it through the chassis and connect it to the VIN connector on the motor board. Finally connect the negative battery wire to a GND connect on the motor board.

According to the technical specifications for the kit, 4.5 – 9.0 volts is required to

power the motor control board so we can use either 4 X AA Alkaline = 6.0 V, or 4 X NIMH rechargeable = 4.8 V batteries. It is also possible to power motors separately, and the specs say that 4.5 to 36 Volts can be used. This range may be fine for the board, but for these motors 10.0 Volts is the maximum, with 6 Volts being the optimum, but they run fine on 4.8V..

If you are using only one power supply for both board and the motors then ensure that the jumper, which is by the white power switch, is across the pins marked VIN and VM. If you are going to use a second power supply for the motors then change the jumper so that it is across VM and NC and connect the battery to blue connecters VM and GND.



So that completes the construction of the car, now for the Apps that will control it.

Software the Android application and the Car application.

There are two application that are used to control the car. The Android application that takes the human input, from the selection of the arrows on the screen, and passes

these onto the Car application that in turn controls the wheel speed and direction of rotation.

The android device that I will be using is a Nexus 7, and so I downloaded the app "car(ap mode)" from the Google Play Store app on the Nexus, if you are using a different Android device that does not have Play Store app then you can download the app from https://play.google.com/store/apps/details?id=com.remote.yingwen.carsochet

Now a connection needs to be established between the Android device and the Smart Car. The application that should be on the Smart Car will set up its own network with a SSID of "DoitWiFi". So first switch on on the Smart Car by pressing the I side of the power switch in the middle of the car. You should see a single blue LED on in the middle of the ESP8266-12E board and a green LED on the motor board. No LED? Then check the white power switch on the motor board, it should be in the power, ON, position. Still no LEDs, and you have good batteries in the battery box, then you need to follow the power connection through from the battery box to the Motor board with a volt meter to see where the problem is.

So now go to the "Settings-WIFI" page on your device, and you should see the "DoitWIFI" network available for connection. Select the DoitWIFI and then enter the password "12345678' and then select connect. On the Nexus you will see that you have connected, but there is no internet, this is correct.
But, if when you go to the "Settings-WIFI" page on your device you do not see the "DoitWIFI" network available for connection, but you see "Doit_ESP_14054915" or something similar then the wrong program is loaded into the Cars ESP8266-12E. You may also notice that the blue LED on the ESP8266-12E is flashing and not on steady as it should be. Please go to the section titled "Managing the ESP8266-12E software" below, to see how to load the correct application.

Now you can start the "DOIT Car" app that you previously down loaded to your android device, and it should come up with a page of arrows as shown in Step 4 of the Instructable line above, with the words "Connect success" at the top of the screen.

I would suggest that you lift the car off the ground, I put mine on a couple of blocks, before carrying on. The next task is to check that we have the motors wired up correctly. From the DOIT Car app press the single arrow at the top of the screen, this should make the car go forward. In my case the left hand motor went forwards, but the right hand motor went backwards. So I need to reverse the connections for the right hand motor. So switch off the car battery and make any necessary changes. Now you can test again, and if all is well try it on the ground. If it the Car is going as you would like then fine, but YOU have the power to change how the car runs, as can be seen in the next section.

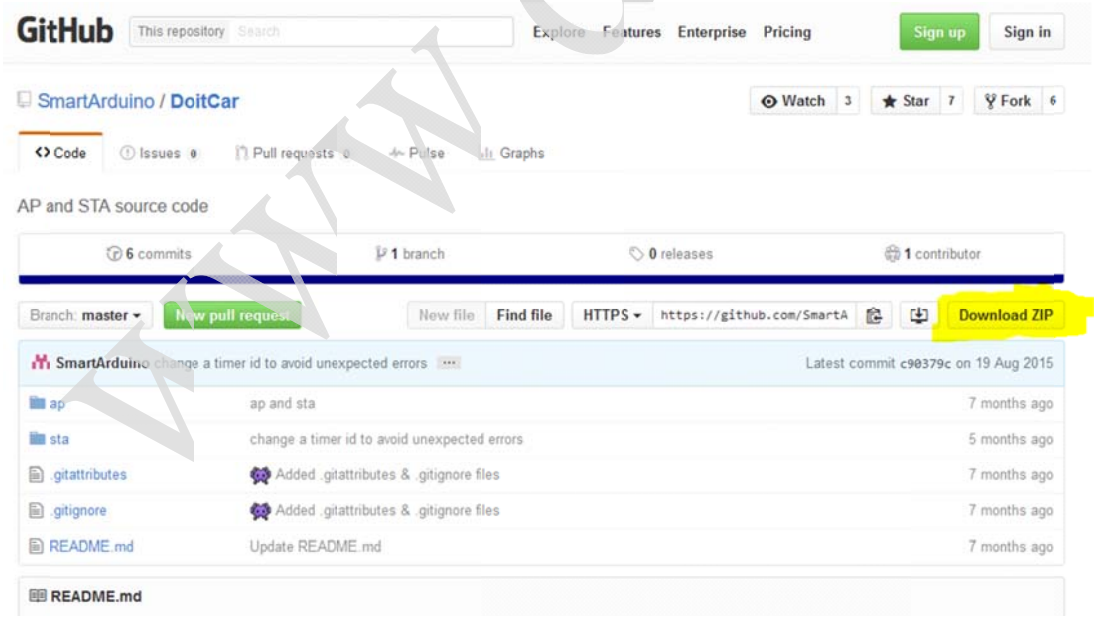
Changing the software in the Car.

There are various languages available for programming the ESP8266, the Car application is written in LUA script, which is very similar to other high level languages such as Python and that used by the Arduino IDE. It is also possible to program the ESP8266 from the Arduino IDE but that is not covered here, but you can find an instructable here : http://www.instructables.com/id/Programming-the-ESP8266-12E-using-Arduino-software/?ALLSTEPS

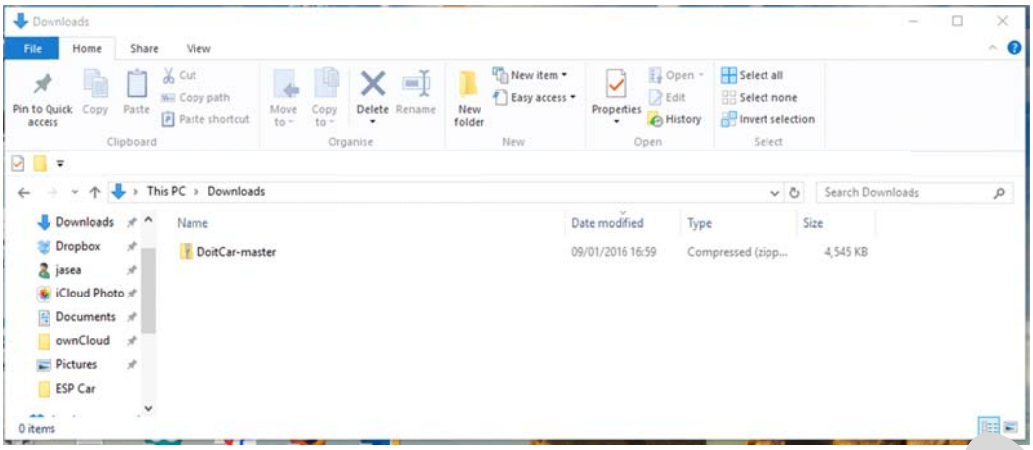
Managing the ESP8266-12E software

SmartArduino have made available the basic software that can be installed in the ESP8266-12E on the Car, but you will need some additional software in order to upload and if necessary make changes to this code. When the ESP8266 powers up the firmware, NodeMCU, looks for a file called init.lua to run. Some tutorials call all of their examples init.lua so that they automatically start, but the correct way is to have init.lua call the file that you want to run. This why we have two files to process.

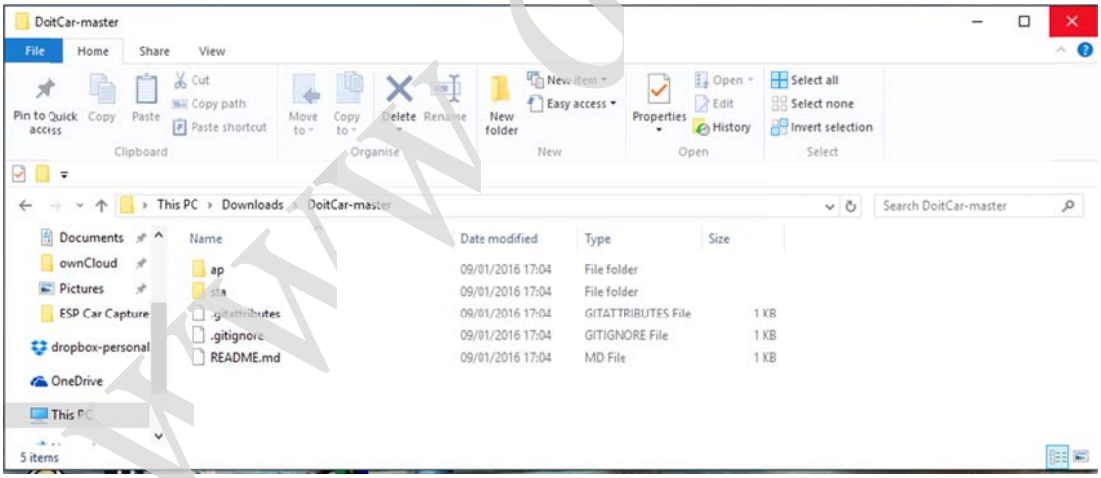
The code can be found here: https://github.com/SmartArduino/DoitCar In the picture below you will see a button “Download ZIP” highlighted selecting this button will download the code we are after.



The ZIP file will be found in your “Downloads” directory, and you will need to select with the right mouse button and then select “Extract All” and then “Save” on the next Pop Up.



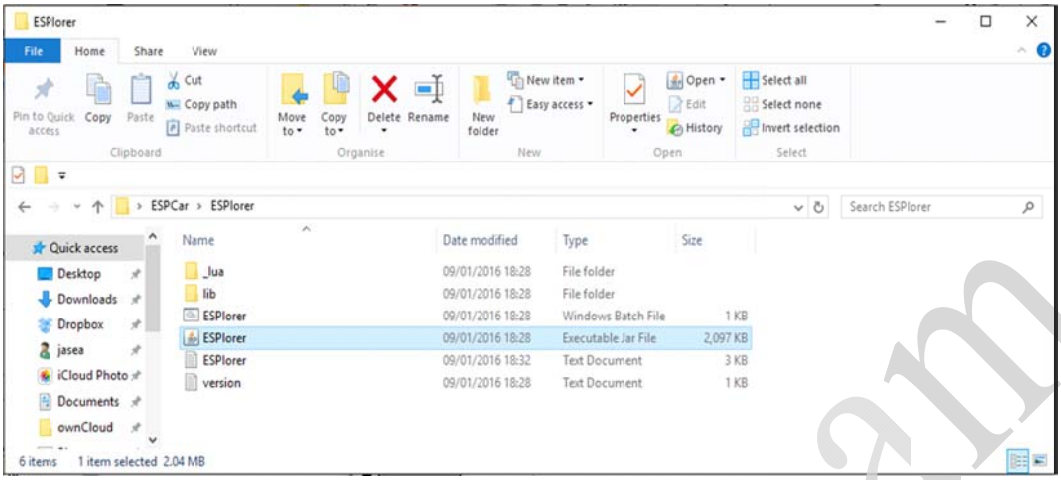
You should now have the DoitC-master.zip file and a DoitCar-master directory in your Download directory. Select DoitCar-master directory and you should see the following,



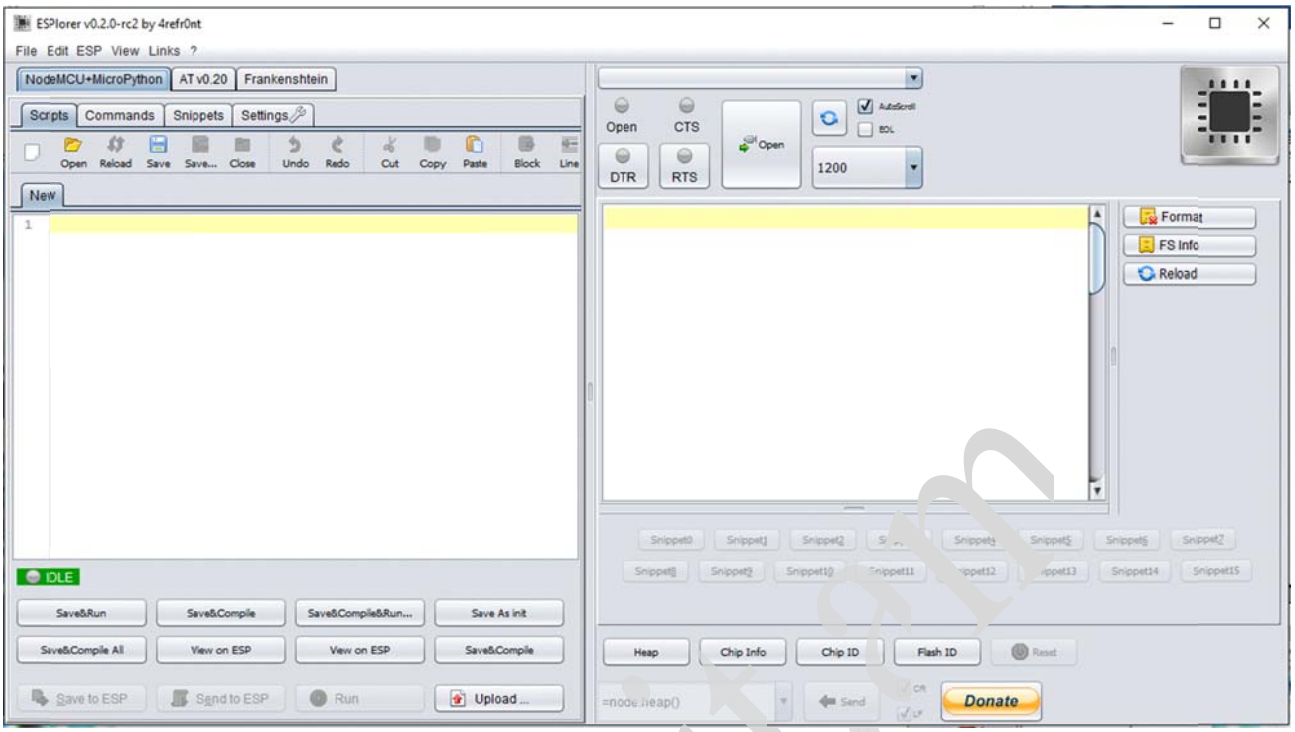
At this point it time all we are interested are the files in the AP directory. I created a folder on the desk top called ESP Car and copied this AP directory into it.

The Integrated Development Enviromnent (IDE) that is used for the ESP8266-12E is called the ESPlorer IDE and can be found here: http://www.esp8266.com/viewtopic.php?f=22&t=882
Halfway down this post you, under the heading “Tutorial” you will find a document by Rui Santos that will tell you how to install ESPlorer and where to get the current level of Java that is required.

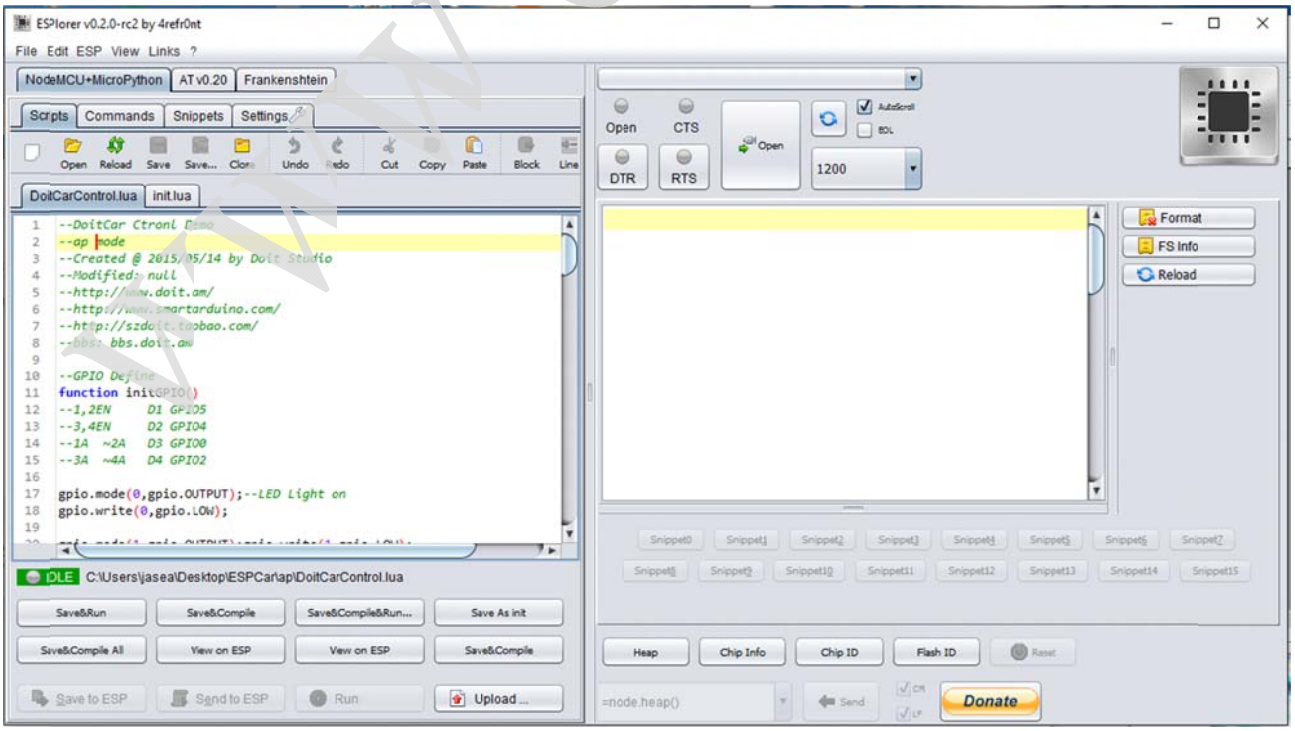
After downloading and expanding the zip file, I copied the ESPlorer directory into my ESPCar directory. To start ESPlorer double click on the ESPlorer Jar file.



When esplorer starts it will look like this:

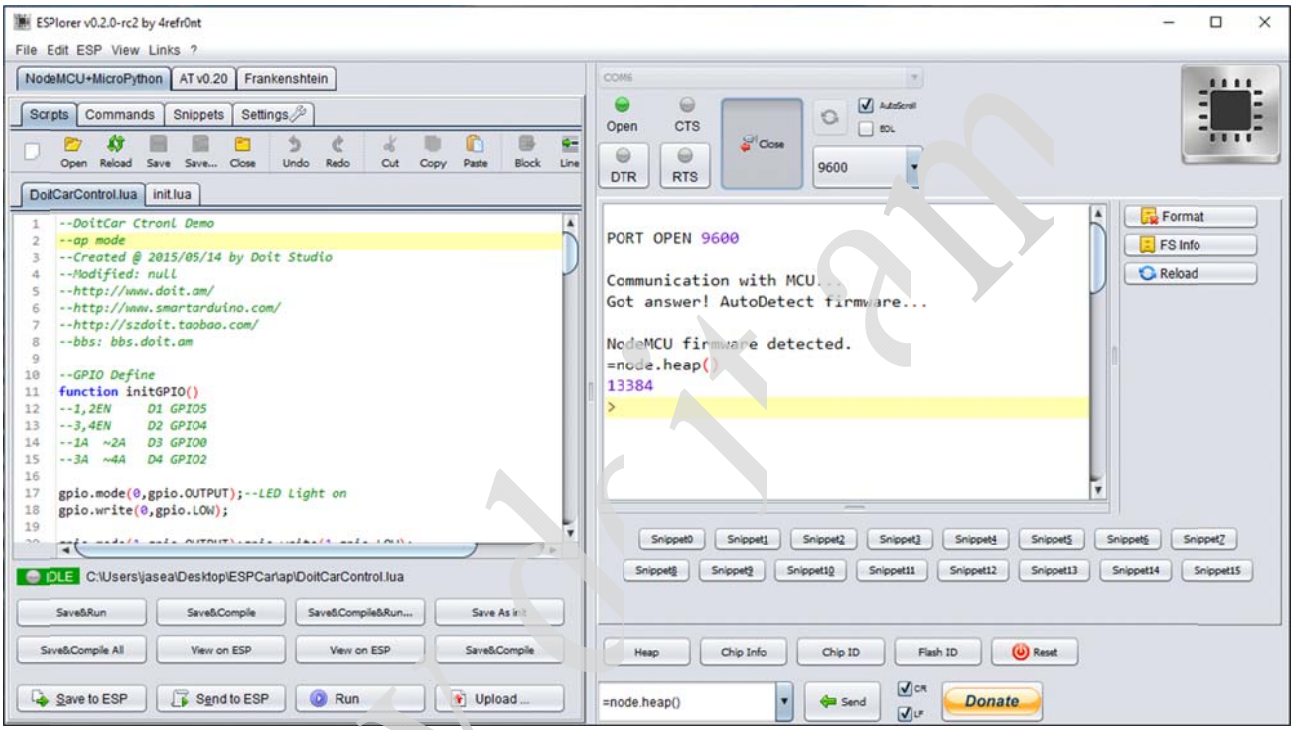


Now the two source files DoitCarControl.lua and init.lua in the AP directory are loaded into ESPlorer, by using File-Open from disk and then selecting the directory from the Look in pull down menu and then selecting DoitCarControl.lua and then open. Repeat this file open for the init.lua and then the ESPlorer screen should look like this:



You can see either of the two files by selecting the tab with its name on it. Check that both files have “--ap mode” on the second line.

To load these programs onto the ESP8266-12E , first remove it from the Motor Driver board and then connect it to PC using a micro USB lead. Press the Refresh button, this is the button to the right of the open button, in the picture above, with two round arrows on it. The port ID that the ESP8266 is connected to will now appear in the pull down menu above the Open button If there is more than one then select the correct port for the ESP8266. Now press the Open button.



Pressing the “Save to ESP” button will now send the DoitCarControl.lua file down to the ESP8266.
Now select the Init.lua tab and the “Save to ESP” button to complete the process.

If you went through this process to overcome the incorrect network name you can now close down ESPlorer and remove the ESP8266 USB connect and replace the ESP8266-12E onto the Motor Driver board and retry connecting to the “DoitWIFI” network from your Android device. Note that the end of the ESP8266 with the USB socket fits nearest the blue connectors.

Now that you know how load up the ESP8266, you can make changes to the source to get the car to do what you want.

First notice in the right hand frame of the picture above, the output from the NodeMCU firmware is shown. In the DoitCarControl.lua there is section that handles the input from the Android application, and it outputs information to this window. So you can see that when you select the forward arrow on your Android device a message TCPServ: 1 is shown in the window. And that at line 89 of DoitCarControl.lua this command is processed by setting both motors in the forward

direction.

Have fun playing with the code and learning about LUA programming, don't forget if you get into problems, you can always downlad the original code and start over again.

Support and Service

Forum: http://bbs.smartarduino.com/

How to order it: http://www.smartarduino.com/view.php?id=94572