LaurTec

Progetto di un Robot comandato WiFi

Car Control Lua

Autore : *Paolo Salvagnini*

ID: UT0011-IT

INFORMATIVA

Come prescritto dall'art. 1, comma 1, della legge 21 maggio 2004 n.128, l'autore avvisa di aver assolto, per la seguente opera dell'ingegno, a tutti gli obblighi della legge 22 Aprile del 1941 n. 633, sulla tutela del diritto d'autore.

La mancata osservanza della legge 22 Aprile del 1941 n. 633 è perseguibile con la reclusione o sanzione pecuniaria, come descritto al Titolo III, Capo III, Sezione II.
A norma dell'art. 70 è comunque consentito, per scopi di critica o discussione, il riassunto e la citazione, accompagnati dalla menzione del titolo dell'opera e dal nome dell'autore.

AVVERTENZE

I progetti presentati non hanno la marcatura CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.

L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.
Si fa inoltre presente che quanto riportato viene fornito così com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Indice

Introduzione....4
Applicazioni....4
Analisi del Progetto....5
Utilizziamo l'editor Arduino....13
Analisi finale....19
Bibliografia....21
History....22

Introduzione

Devo fare una premessa prima di presentarvi questo lavoro che è in assoluto il mio primo lavoro in ambito WiFi. Esso nasce dal suggerimento del collega Piero ad usare una scheda che egli reputa interessante, basata sul transceiver WiFi ESP8266 della società Espressif. Essendo il primo lavoro non solo con il transceiver ESP8266 ma anche con i linguaggi di programmazione richiesti per lo sviluppo delle applicazioni, inizialmente volevo solo condividere il programma relativo alla mia applicazione, che secondo me aveva qualche spunto interessante. Mauro invece mi ha chiesto un articolo di quelli degni di esser pubblicati. Purtroppo l'argomento è talmente vasto che per raccontare i dettagli del transceiver ESP8266 sarebbe necessario un libro e altrettanto dicasi per i linguaggi di programmazione utilizzati. Per questo motivo mi sono quindi posto l'obbiettivo di raccontarvi come sono giunto alla fine di ciò che ho chiamato Car Control Lua, senza entrare nei dettagli di ogni aspetto tecnico.

Applicazioni

Le applicazioni che si possono realizzare seguendo questo articolo, sono veramente molteplici:

- Robot telecomandati da cellulare o tablet.
- Controllo remoto di carichi elettrici.
- Controllo a distanza di un antifurto.
- Applicazioni domotiche.

Analisi del Progetto

Di schede che montano il transceiver ESP8266 ne esistono ormai diverse versioni che si differenziano essenzialmente per il modo in cui si programmano e per il numero delle porte connesse ai pin del connettore, spesso presente sulla scheda stessa. Storicamente parlando, le prime schede sono nate solo con un'interfaccia seriale e necessitano quindi, per la loro programmazione, di un convertitore USB-UART. Io ho invece preferito la versione ESP8266 12E DEV KIT (Figura 1) che sebbene di dimensioni maggiori ha già montato un convertitore USB-UART e un regolatore di tensione da 5V a 3.3V. La tensione di funzionamento dell'ESP8266 è infatti di 3.3V e quindi la presenza del regolatore fa si che la scheda possa essere connessa direttamente alla porta USB del PC che fornisce infatti 5V.

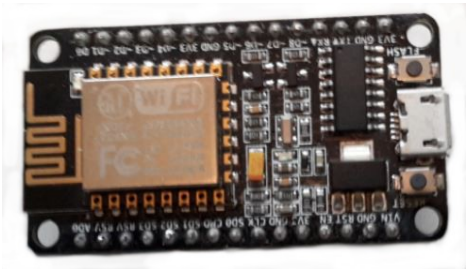

Figura 1: *Modulo ESP8266 12E DEV KIT.*

Prima di connettere la scheda con l'ESP8266 alla porta USB bisogna tenere presente che la corrente assorbita dall'ESP8266 in trasmissione, raggiunge i 200mA, perciò bisogna prima fare un calcolo della corrente totale del vostro Hardware. Per tale ragione è bene prendere anche in considerazione di usare un'alimentazione esterna.
Detto questo, bisogna installare il driver per il convertitore USB-UART montato sulla scheda ESP8266, che nel mio caso e credo, nella maggioranza delle schede cinesi, è un CH340. Questo driver è generalmente disponibile sul sito del venditore della scheda. Il sistema operativo Windows 8, cercando il driver in rete potrebbe anche installare la versione ufficiale, senza richiedere dunque di scaricare ed installare un driver esterno.
Poiché il venditore cinese dal quale si possono comprare le schede di sviluppo a pochi euro, si rifiuta di dare un qual si voglia supporto e non avendo informazioni sulla scheda, ho pensato che non vi fosse un firmware installato. Questo in generale non è vero ma saper aggiornare il Firmware interno alla scheda è comunque un passo da imparare visto che sono presenti in rete diverse opzioni. In questo articolo ho deciso che sarebbe stato meglio partire da zero. In particolare in base alla scheda acquistata si potrebbero avere diversi casi, tra cui i più probabili sono:

- L' ESP8266 è programmato per supportare comandi AT.
- L' ESP8266 è programmato con il firmware nodemcu.
- L' ESP8266 non è programmato, eccetto il bootloader presente di default.

In particolare il Firmware nodemcu permette di interpretare programmi scritti nel linguaggio di programmazione Lua, che utilizzerò negli esempi che seguiranno. Il linguaggio di programmazione Lua permette di programmare l'ESP8266 ad un livello piuttosto astratto senza dover entrare nei dettagli dell'architettura del transceiver e della MCU integrata.

Capite le varie opzioni disponibili, vediamo come caricare il nostro Firmware. Per fare questo, serve un software idoneo, la Espressif fornisce un software ad hoc con molte opzioni, che rendono il suo utilizzo piuttosto complesso, per questo ho scelto un'altra applicazione nominata Flasher, scaricabile dal sito:

```
https/github.com/nodecpu/nodecpu-flasher
```

In particolare è disponibile sia la versione per 32 che 64bit. Sebbene più lento di altri programmatori, Flasher è di estrema facilità di utilizzo e non necessita di installazione. Eseguendo il file ESP8266flasher.exe vi apparirà sullo schermo la finestra di Figura 2 dalla quale potete selezionare la porta seriale utilizzata dalla scheda collegata al PC.

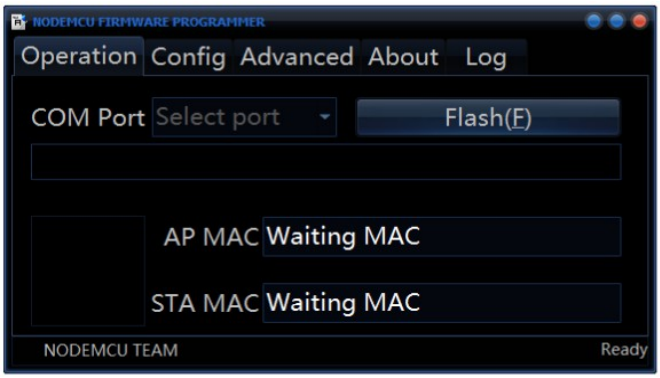

Figura 2: *Finestra iniziale del programma Flasher.*

Cliccando su CONFIG otterrete la finestra di Figura 3, cliccando sul bottone della prima fila a forma di ingranaggio, quello compreso tra la barra verde e quella blu, potete selezionare il Firmware da scaricare, ovvero:

```
nodemcu_float_0.9.6-dev_20150704.bin
```

che dovrete preventivamente scaricare dal sito nodemcu:

```
https://github.com/nodemcu/nodemcu-firmware/releases
```

Impostate nuovamente il Tab Operation e cliccate su Flash, accertandovi di aver selezionato, nella casella di sinistra di Figura 3, solo la prima riga. La fase di programmazione sarà indicata da barra blu, contemporaneamente potrete prender nota degli indirizzi MAC della scheda.

A operazione compiuta la scheda con ESP8266 è pronta a funzionare per mezzo di un IDE dedicato alla programmazione Lua. In particolare la scelta è stata di usare ESPlorer che è scaricabile dal sito:

```
http://esp8266.ru/esplorer
```

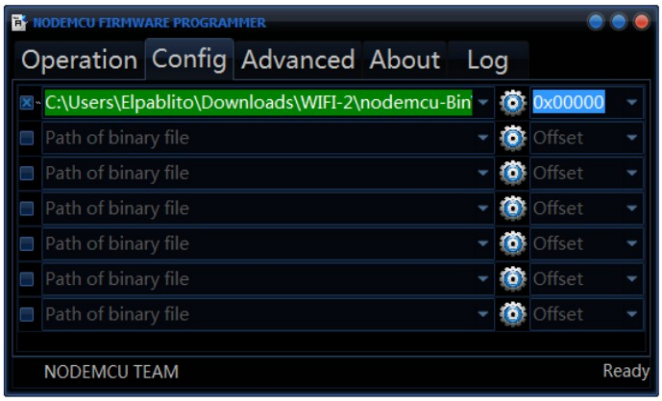

Figura 3: *Schermata per la selezione del Firmware.*

Il sito presenta anche un link per scaricare un eventuale manuale, anche se non esaustivo, ma permette di conoscere le funzioni di base dell'IDE. Molti di voi si chiederanno il perché della scelta dell'IDE ESPlorer. Se leggerete il libro "Kolban's Book on ESP8266" troverete che esistono IDE che vanno da Eclipse all'IDE di Arduino e altrettanti linguaggi con cui programmare.
In particolare la Espressif fornisce un ambiente di sviluppo software (SDK) basato su Eclipse e programmazione C, che permette di utilizzare, al prezzo di una maggiore complessità di programmazione, tutte le funzioni del transceiver ESP8266. Sarà quindi una scelta futura di ognuno di voi cosa adoperare. Io ho scelto il linguaggio di programmazione Lua e l'IDE ESPlorer perché non necessitano di particolari conoscenze tecniche e risultano sufficientemente intuibili per essere usati al primo approccio.

Passiamo ora finalmente al programma. Questo è pensato per far funzionare una macchinina o robot che dir si voglia. Pertanto esistono i comandi di avanti, indietro, ferma, destra, sinistra e il controllo di velocità. In effetti, a parte i nomi, che possono essere facilmente modificati, si hanno a disposizione dei comandi analogici e dei comandi digitali applicabili a una qualsiasi interfaccia per comandare ciò che si vuole, come per esempio accendere e spegnere dei carichi da remoto.
In particolare non vi è un programma residente sull'elettronica (PC, Tablet, telefonino) che andrà a connettersi con ESP8266. È infatti il transceiver ESP8266 che interrogato, genera la grafica necessaria sul display del chiamante. In gergo, il modulo ESP8266 si comporta come un web server che genera, ovvero fornisce, il codice HTML per permettere a un qualunque browser, indipendentemente dal sistema operativo e hardware (PC , table cellulare), di poter accedere al sistema.

Con la scheda ESP8266 connessa alla porta USB lanciate ESPlorer. Premete quindi il pulsante OPEN. Nel riquadro di destra, vi comparirà un log con le informazioni dell'avvenuta connessione, come visibile nella Figura 4.
Prima di caricare il file del programma vi suggerisco di fare il seguente esercizio. Copiate le righe sottostanti nel riquadro di sinistra di ESPlorer.

```
wifi.setmode(wifi.STATIONAP)
wifi.sta.config("WIFI-2.4-CA4D","xxxx")
wifi.sta.connect()
print(wifi.sta.status())
ip=wifi.ap.getip()
```

```
print(ip)

-- print ap list
function listap(t)
 for k,v in pairs(t) do
     print(k.." : "..v)
 end
end

wifi.sta.getap(listap)
```

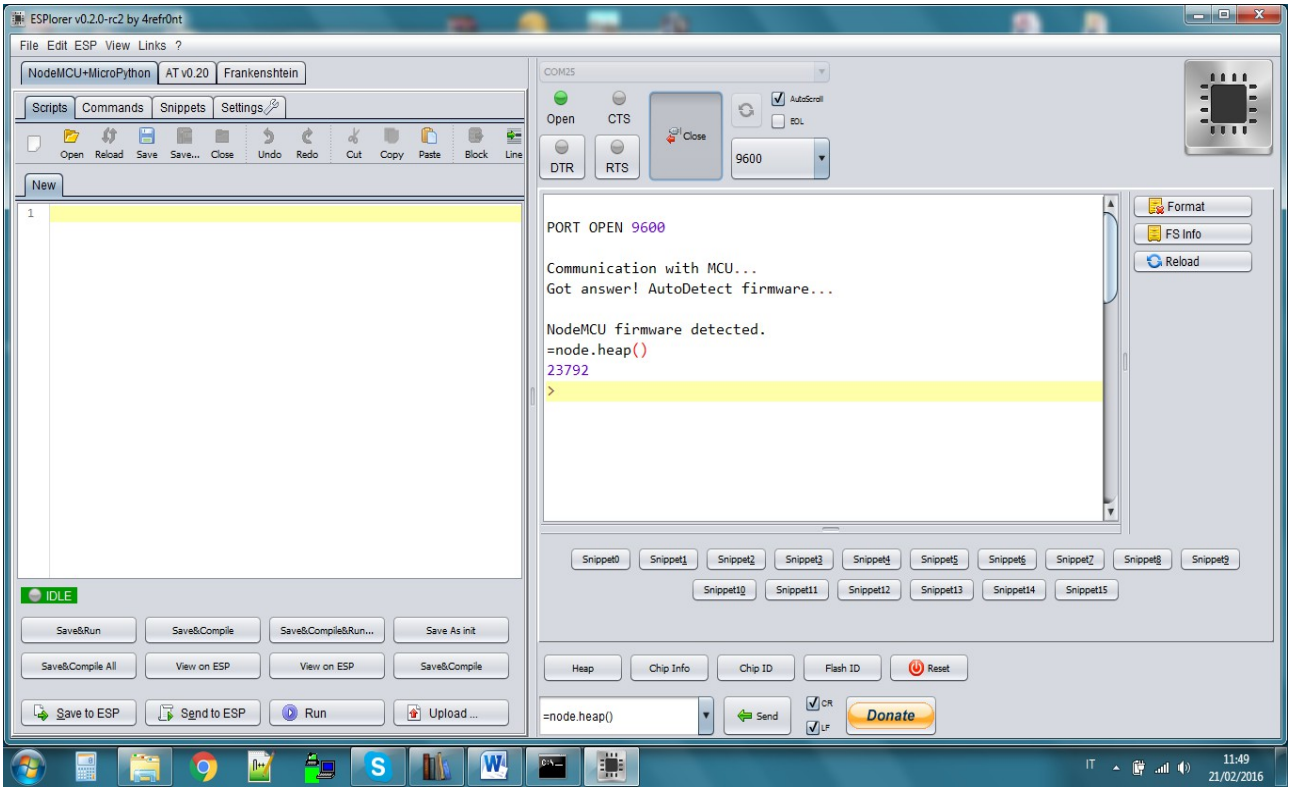

Figura 4: *Schermata di avvio dell'applicazione ESPlorer.*

Selezionate le prime tre righe ove al posto di WIFI-2.4-CA4D avrete digitato il nome della vostra WiFi e al posto di xxxx la password del vostro router. Premete ora il pulsante in alto a destra con scritto Block. Cosi facendo avrete creato un oggetto che può funzionare come Station o come Access Point e che sta connettendosi alla vostra rete WiFi. Selezionate la quarta riga e premete Block dovrebbe comparirvi un 3 sulla destra che identifica lo stato della station. Selezionate la quinta e la sesta riga e premete Block, sulla destra vi comparirà l'indirizzo IP della scheda, prendetene nota visto che vi servirà per accedere da remoto al webserver creato dalla scheda. Infine selezionate :

```
function listap(t)
 for k,v in pairs(t) do
     print(k.." : "..v)
 end
end
```

Premendo Block avrete in risposta tutte le connessioni possibili viste dal vostro Access Point. In basso al centro vi è una casella con un menù a tendina dove di default è scritto

`node.heap()` e al suo fianco un bottone con scritto *Send*. Selezionate qualcosa, ad esempio:

```
=wifi.ap.getip()
```

e quindi premete *Send* ed osservate la risposta. Vi suggerisco di provare anche le altre opzioni della finestra a tendina. Questa esperienza vi fornirà indirettamente maggiori informazioni sui comandi che potete usare nel vostro programma Lua.
Premete ora il tasto *Format* (tasto sulla destra) e aspettate fino a quando, la formattazione sarà terminata. Nella finestra di Log comparirà la scritta *format done*. La formattazione fatta a questo livello non cancella il Firmware precedentemente caricato ma solo quell'area di memoria flash all'interno dell'EPS8266 utilizzata dal Firmware per caricare il programma Lua da interpretare. In particolare il file principale che l'interprete Lua nodemcu si aspetta, si chiama *init.lua*. Siamo finalmente giunti all'applicazione.

L'Hardware adoperato per testare il progetto prevede sei LED connessi con una resistenza da 470 ohm verso massa e connessi con il positivo ai pin D1, D2, D3, D4,D5 e D6 (Figura 5).

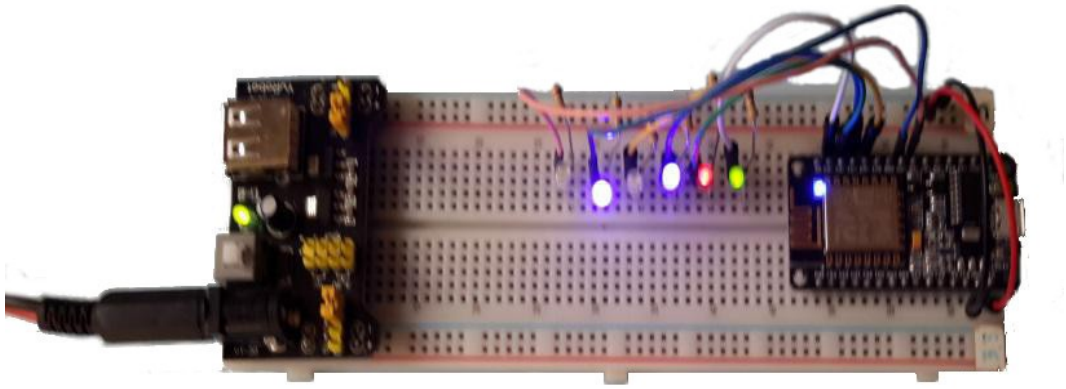

Figura 5: *Dettaglio del sistema usato per la verifica del programma.*

Caricate il file init.lua con *open from disk* e sostituite nella seconda riga del codice il nome del vostro router WiFi e la corrispondente password come nel caso precedente.

```
wifi.setmode(wifi.STATION);
wifi.sta.config("belkin.44e4","xxxx");
print(wifi.sta.getip());
wifi.setmode(wifi.STATIONAP);
wifi.ap.config({ssid="Test",pwd="12345678"});
```

Con le ultime due righe l'ESP8266 viene configurato come un Access Point con il nome Test e pass "12345678". Naturalmente potete sostituire nome e password a vostro piacimento. Questo fa si che localmente comparirà una nuova rete WiFi dal nome (SSID) Test. Avete quindi la possibilità di connettervi alla scheda con l'ESP8266 tramite il vostro WiFi adoperando l'indirizzo IP precedentemente annotato oppure connettendovi direttamente all'Access Point Test. Se non avete annotato l'indirizzo IP che il router fornisce al modulo ESP8266, selezionate il comando `=wifi ap.getip()` e quindi premete *Send*.

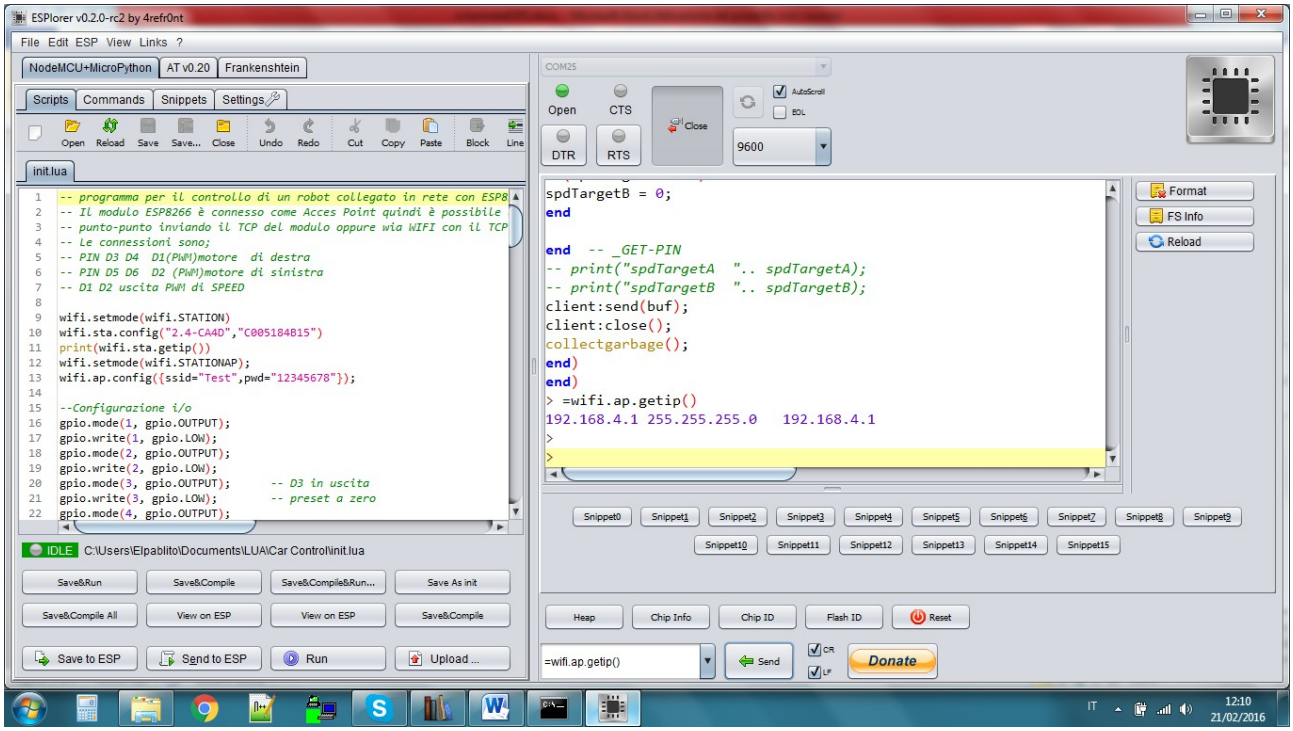

Figura 6: *Dettaglio del programma caricato nell'IDE ESPlorer.*

Connettete il vostro Tablet con la nuova rete WiFi Test e collegatevi dal vostro browser all'indirizzo IP della scheda. Quale risposta dovrà aprirsi sul monitor una pagina HTML come quella che si vede in Figura 7.

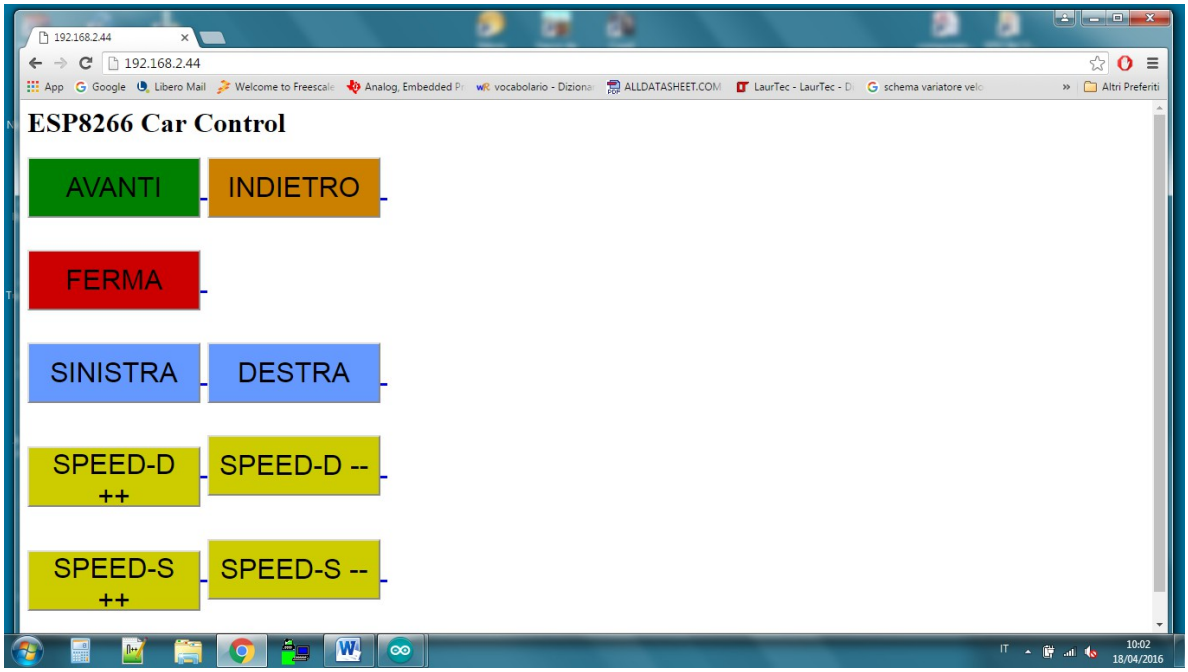

Figura 7: *Pagina principale inviata dal modulo ESP8266.*

Non vi resta che premere i vari pulsanti per vedere cosa succede.

Aprendo il file .lua, meritano alcune considerazioni le righe sottostanti. Questo gruppo di istruzioni sono quelle che generano la grafica sul vostro monitor e che sono state oggetto di un po' di grattacapi poiché scritti in lingua non a me nota. È quindi doveroso capire almeno come poterle modificare, salvo l'approfondimento del linguaggio da coloro a cui interessa.

```
--Intestazione
buf = buf.."<h1> ESP8266 Car Control<h1>";

buf = buf.."<a href=\"/?pin=ON1\"> <button style=\"background-color:
#008000;width:200px;height:70px\"> <font size=\"6\">AVANTI </font>
</button> </a>";

buf = buf.."<a href=\"/?pin=ON2\"> <button style=\"background-color:
#cc8000;width:200px;width:200px;height:70px\"> <font size=\"6\">INDIETRO
</font> </button> </a> <br/><br/>";

buf = buf.."<a href=\"/?pin=ON5\"> <button style=\"background-color:
#cc0000;width:200px;height:70px\"> <font size=\"6\">FERMA </font>
</button> </a> <br/><br/>";

buf = buf.."<a href=\"/?pin=ON3\"> <button style=\"background-color:
#6699ff;width:200px;height:70px\"> <font size=\"6\">SINISTRA</font>
</button> </a>";

buf = buf.."<a href=\"/?pin=OFF3\"> <button style=\"background-color:
#6699ff;width:200px;height:70px\"> <font size=\"6\">DESTRA</font>
</button> </a> <br/><br/>";

buf = buf.."<a href=\"/?pin=ON4\"> <button style=\"background-color:
#cccc00;width:200px;height:70px\"> <font size=\"6\">SPEED-D ++ </font>
</button> </a>";

buf = buf.."<a href=\"/?pin=OFF4\"> <button style=\"background-color:
#cccc00;width:200px;height:70px\"> <font size=\"6\">SPEED-D -- </font>
</button> </a> <br/><br/>";

buf = buf.."<a href=\"/?pin=ON6\"> <button style=\"background-color:
#cccc00;width:200px;height:70px\"> <font size=\"6\">SPEED-S ++ </font>
</button> </a>";

buf = buf.."<a href=\"/?pin=OFF6\"> <button style=\"background-color:
#cccc00;width:200px;height:70px\"> <font size=\"6\">SPEED-S -- </font>
</button> </a> <br/><br/>";
```

`buf=buf..` significa che il buffer è quello precedente concatenato, tramite i due punti , alla stringa successiva contenuta tra " ".
Ogni comando e/o istruzione è compreso tra due TAG es: `<h1>.....<h1>` . Questo TAG definisce un'intestazione. L'intestazione è a vari livelli a scendere, partendo da `<h1>` fino a `<h6>`, scendendo di livello diminuisce anche la dimensione del carattere. La prima riga del buffer fa comparire l'intestazione ESP8266 Car control.

I due TAG `<br/><br/>` servono invece ad andare a capo. I nostri pulsanti saranno quindi affiancati sulla stessa riga fino a quando non troveremo questi due TAG. Di conseguenza i pulsanti AVANTI e INDIETRO si trovano sulla stessa riga, poi si va alla riga seguente con FERMA e nuovamente a capo con due coppie di pulsanti per tutte le righe successive. Analizziamo ora la stringa che crea il pulsante. La prima parte

```
<a href=\"/?pin=ON1\">
```

costituisce la risposta `pin=ON1` che viene inviata dal vostro Tablet al transceiver ESP8266. La parte seguente identifica l'oggetto button e le sue caratteristiche.

```
<button style=\"background-color: #cccc00;width:200px;height:70px\">
```

Colore di fondo:

```
background-color: #cccc00;
```

Lunghezza e altezza del button in pixel:

```
width:200px;height:70px
```

Per finire con il nome che viene scritto all'interno del button e la dimensione del carattere

```
<font size=\"6\">SPEED-S -- </font> </button>
```

La rimanente parte del programma è facilmente intuibile. Alla ricezione di un gruppo di dati questi sono confrontati con i rispettivi valori ON, OFF e di conseguenza modificati i valori in uscita delle porte. Il controllo della velocità, quindi l'uscita analogica, è fatta a scalini, quindi ogni volta che viene premuto il pulsante questa viene incrementata di una quota fissa nel nostro caso 50.

Utilizziamo l'editor Arduino

Realizzato il progetto per mezzo dell'applicazione scritta con il linguaggio Lua, vediamo un altro metodo per raggiungere lo stesso scopo. Ho ricevuto una seconda scheda ESP 12E DEVKIT V2 e ho quindi potuto verificare quanto sospettavo, la scheda viene fornita con il Firmware nodemcu ed è pronta per essere programmata in Lua.
Sebbene la prima parte del progetto ed esempi abbiano fatto uso del linguaggio Lua, ho incontrato varie difficoltà con ESplorer su cui non vorrei soffermarmi. Ciò che mi ha lasciato perplesso, in special modo ai primi passi, è il fatto era che non era chiaro se l'errore fosse mio oppure del Browser. Ho quindi pensato di provare a riscrivere l'applicazione sulla piattaforma Arduino o meglio l'IDE di Arduino riadattato per il transceiver ESP8266 senza dover usare la scheda Arduino. È bene precisare questo, visto che sono presenti anche molti progetti in cui si fa uso della scheda Arduino e dell'ESP8266. In questo caso continuerò a programmare direttamente l'ESP8266.

Usando l'IDE di Arduino dobbiamo di conseguenza seguire le regole del gioco, anche se ad alcuni non piacciono, ma spesso sono molto comode e soprattutto ci permettono di raggiungere lo scopo. Prima di passare ai dettagli vorrei precisare quali sono le differenze con il programma precedente. È stato aggiunto il controllo di un display LCD 16x2 o 16x4 via I2C non per una reale necessità, ma come esercizio di programmazione del micro del ESP8266. A coloro cui non interessa, basta semplicemente che non colleghino il display. La pagina HTML è notevolmente migliorata, con il Lua non ero riuscito a fare ciò che volevo. In termini di funzionamento il controllo della velocità è diventato unico per ambedue i motori e vi è un ADJ che permette di diminuire leggermente la velocità di uno dei due motori, qualora ce ne fosse bisogno. In Figura 8 è riportato il nuovo montaggio del sistema e la visualizzazione della pagina HTML generata sul Tablet connesso all'ESP8266.

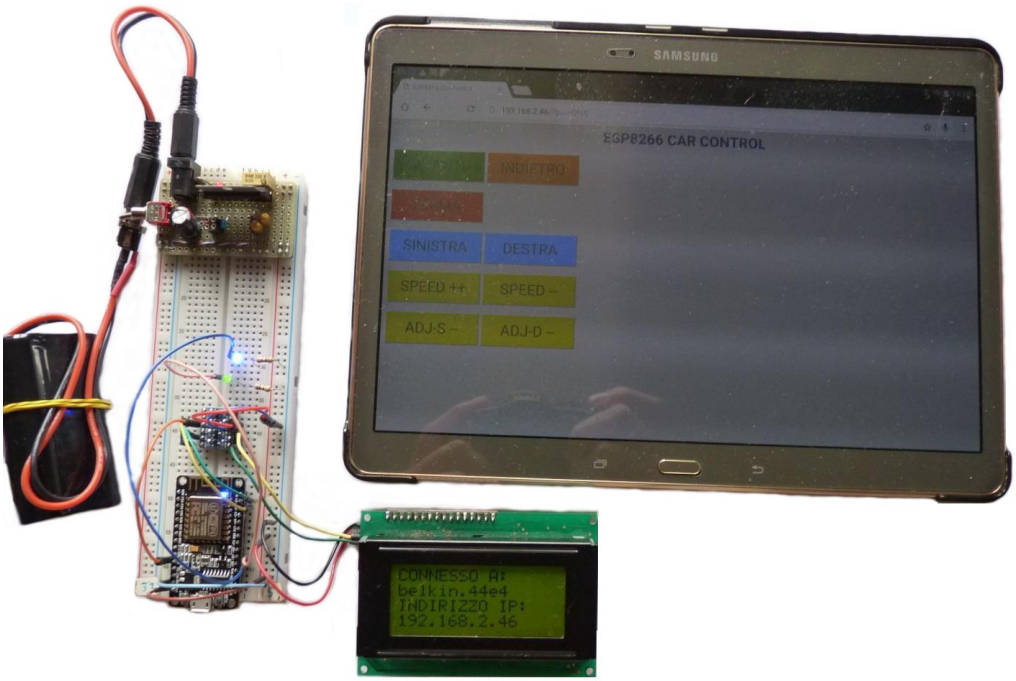

Figura 8: *Sistema programmato con l'IDE Arduino a montaggio completato.*

Per iniziare con il nuovo ambiente di sviluppo, la prima cosa da fare è quella di andare sul sito di Arduino e scaricare il sistema di sviluppo. Installata la versione corrente che è la 1.6.8, dobbiamo configurarla per poter programmare le varie versioni dell'ESP8266. Lanciamo il programma, quindi clicchiamo su *File* e quindi *Impostazioni*.

Ci viene subito chiesto di definire il percorso degli sketch. In Arduino sono definiti sketch i programmi sviluppati che sono del tipo xxx.ino.

Nel mio caso ho definito un percorso del tipo: C:\.....\documenti\ArduinoSketch . Nella parte inferiore della videata nella riga a fianco all'URL aggiuntivi per il Gestore schede, copiate e incollate la riga sottostante:

```
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

confermate e uscite. Cliccate ora su *Strumenti* quindi nel menù a tendina di *Scheda* cliccate su *Gestore Schede*. Andate quindi sull'ultima scheda, quella nominata come: *esp8266 by ESP8266 community*, cliccateci sopra e quindi installatela. A questo punto avrete aggiunto alle schede programmabili dall'IDE anche quelle basate su ESP8266.
Qualora vogliate programmare una scheda quale la mia ESP 12E DEVKIT V2 ritornate in *Strumenti*, *Scheda* e selezionate la scheda *MODEMCU 1.0(ESP12E Module)* altrimenti scegliete la scheda conformemente a quella che andrete a programmare.
La configurazione dell'IDE è terminata.

Chiudete l'IDE di Arduino e andate nella cartella *Arduino Sketch*, creata in precedenza. Create due nuove cartelle, la prima deve avere lo stesso nome del file .ino che in essa copierete e quindi *CarControlWebServerLcd*. La seconda il nome *Libraries* in cui copierete la cartella *LiquidCristal_I2C* che troverete allegata sul sito LaurTec. Riaprite l'IDE andate in *File\Cartella degli sketch* e aprite *CarControlWebServerLcd*. Collegate la scheda ESP8266 all'USB cliccate su: *Strumenti* → *Porta* e confermate la porta seriale che userete. Verificate che non vi siano errori cliccando sul bottone con il Check e scrivete il programma nell'ESP8266 con il bottone con la freccia.
Definita la meccanica di come programmare la nostra scheda, è possibile testarla nello stesso modo precedentemente utilizzato nella prima parte del documento o, in alternativa, potete montare e seguire lo schema di Figura 9.

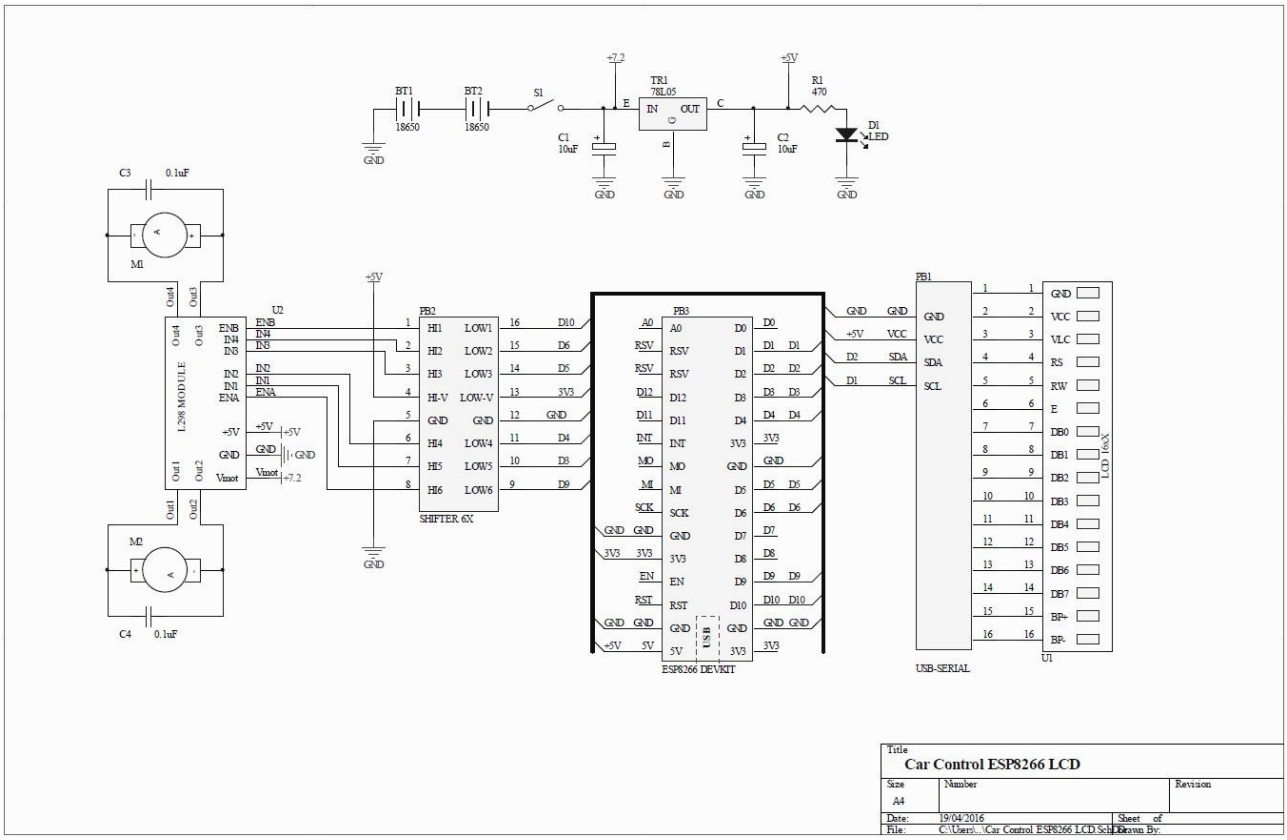

Figura 9: *Schema elettrico per realizzare il robot telecomandato WiFi.*

Il display utilizzato è un 16x4 con connessa una schedina che trasforma i comandi da I2C a parallelo, tramite il solito PCF8574. Naturalmente la schedina è made in Cina e con un costo tale da non valer la fatica di farla. Il connettore è a quattro piedini, due per l'alimentazione e due per la connessione I2C. Vi è anche un potenziometro per la regolazione del contrasto. In Figura 10 è riportato il dettaglio della scheda connessa al modulo LCD.

Figura 10: *Dettaglio della scheda per convertire il protocollo I2C in Parallelo, collegata al modulo LCD.*

Esiste poi una libreria per la scheda, che come avrete già intuito è quella che avete copiato nella cartella *Libraries*. Attenzione alle tensioni. Il display è alimentato a 5V mentre ESP8266 a 3.3V. In effetti i due segnali SDA e SCL sono in uscita quindi potrebbero essere collegati direttamente, però il PCF8574 riconosce come segnale a

livello alto un segale pari a 0.7Vcc quindi 3,5V. Per evitare qualsiasi problema io ho inserito un *level shifter* come riportato nello schema di Figura 11.

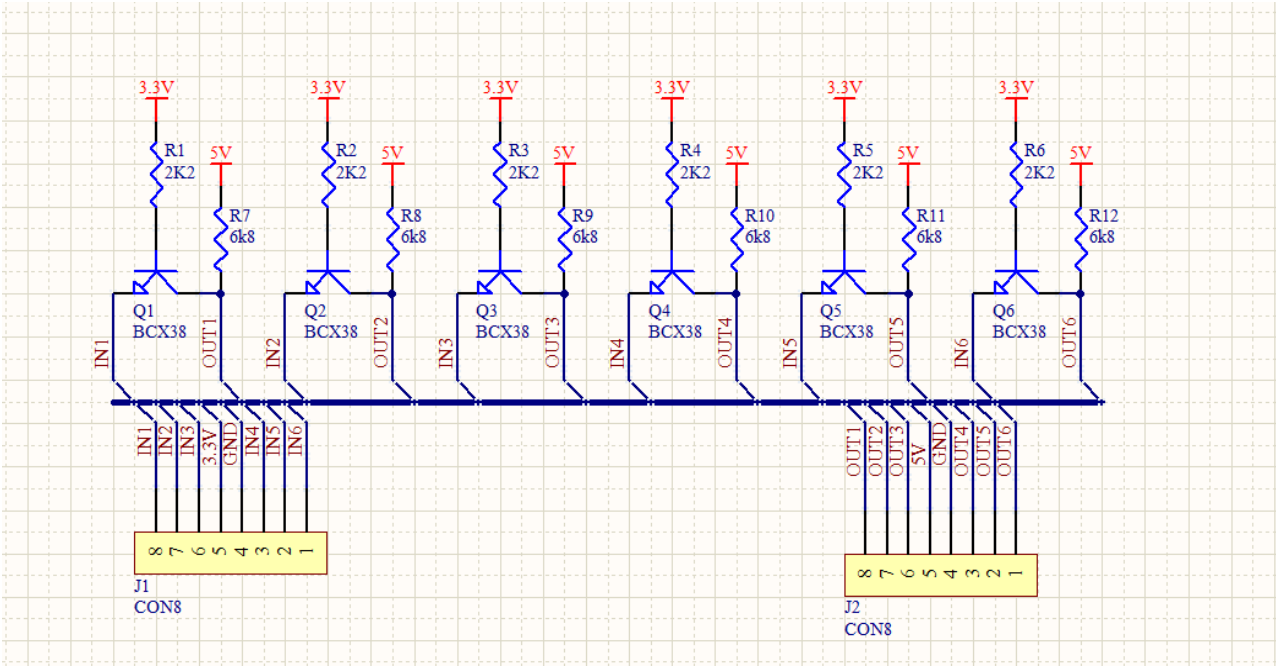

Figura 11: *Schema elettrico con il dettaglio del Level shifter.*

Il traslatore di livello è unidirezionale, e lo ho inserito anche tra l'ESP8266 e il circuito del *motor driver* L298 usato per il controllo dei motori. Il tutto è montato su mille fori ma farò un PCB quanto prima. L'ultimo pezzo di Hardware è il circuito di controllo dei motori con un L298. Anche per questo esistono varie versioni di schede equivalenti in termini di funzionalità, anche se montano integrati diversi. Per motivi di disponibilità io ho usato la scheda di Figura 12.

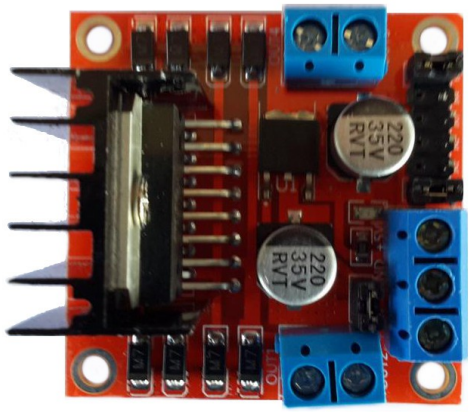

Figura 12: *Scheda per il controllo dei motori.*

Tornando al software scritto, è necessariamente in stile Arduino, ma è scritto in C/C++, quindi più facilmente gestibile. Dopo una parte iniziale in cui si definisce l'oggetto display e le variabili del caso, il resto del programma, ampiamente documentato, si dovrebbe capire senza molte difficoltà visto che rimane comunque piuttosto simile a quello scritto con il linguaggio Lua.

```
// Verifica lo stato fino a connessione avvenuta.
 while (WiFi.status() != WL_CONNECTED) {
    lcd.setCursor(0, 0);
    lcd.print("  CONNESSIONE   ");
    lcd.setCursor(0, 1);
    lcd.print("   IN CORSO     ");
    Serial.print(".");
    delay(500);
 }

 DisplayClear();
 lcd.setCursor(0, 0);
 lcd.print("CONNESSO A:     ");
 lcd.setCursor(0, 1);
 lcd.print(ssid);
 lcd.setCursor(0, 2);
 lcd.print("INDIRIZZO IP:   ");
 lcd.setCursor(0, 3);
 lcd.print(WiFi.localIP());

 Serial.print(WiFi.localIP());

 // Start del server
 server.begin();
```

In questa porzione di programma ho lasciato volutamente le due istruzioni in colore rosso per permettere di identificare l'IP con cui chiamare il server, qualora il display non fosse installato. In questo caso nella finestra del monitor seriale compariranno una serie di puntini ogni 500 ms fino alla connessione avvenuta, mostrando poi l'indirizzo IP del server.

Qualche altra parola si può spendere per la stringa che costituisce la pagina HTML. Riporto solo la parte della stringa che si differenzia da quella descritta nella prima parte. Il 200 nella prima riga significa che la richiesta del cliente è stata ricevuta ed esiste risposta. Osservate i vati TAG concatenati e la differenza tra il TAG di apertura e quello di chiusura.

```
String buff = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n";
buff += "<head>";
buff += "<title> ESP8266 Car Control </title>";
buff += "<style>";
buff +=   "body { background-color: #cccccc; font-family: Arial,
Helvetica, Sans-Serif; Color: #000088; }";
buff += "</style>";
buff += "</head>";
buff += "<body>";
buff += "<h1 ALIGN=CENTER>ESP8266 CAR CONTROL</h1>";
```

La stringa `<title> ESP8266 Car Control </title>` fa sì che sulla cornice della scheda, in alto a sinistra, compaia la scritta contenuta tra i TAG, ovvero il titolo.

```
"<style>"
"body { background-color: #cccccc; font-family: Arial, Helvetica, Sans-
Serif; Color: #000088; }"
 "</style>"
```

Permette di definire il colore di fondo della scheda, i tipi di caratteri usati e il colore delle scritte

```
"<h1 ALIGN=CENTER>ESP8266 CAR CONTROL</h1>"
```

Allinea al centro la scritta seguente.

Dopo aver programmato l'ESP8266 connettetevi col vostro browser al server, tramite l'IP che avrete trovato e vi comparirà la schermata come in Figura 13.

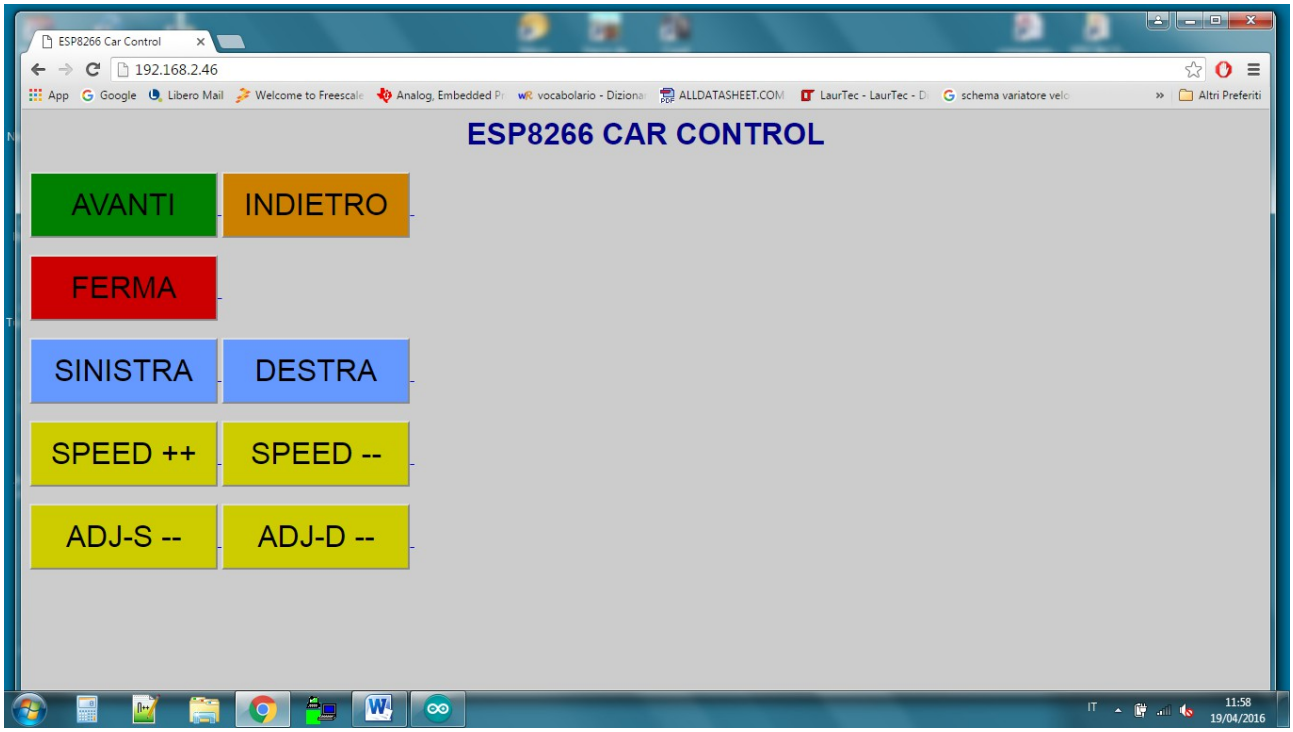

Figura 13: *Finestra mostrata alla connessione con il web server.*

Analisi finale

In Figura 14 è mostrato il Robot sottoposto a test, non è ancora in versione finale, ma il funzionamento è tale da giustificare un rimaneggiamento con un miglioramento del look. Spero di aver incuriosito qualcuno ad usare questa scheda, che sembra avere ampi margini di sviluppo e auguro nuovamente a tutti un buon lavoro.

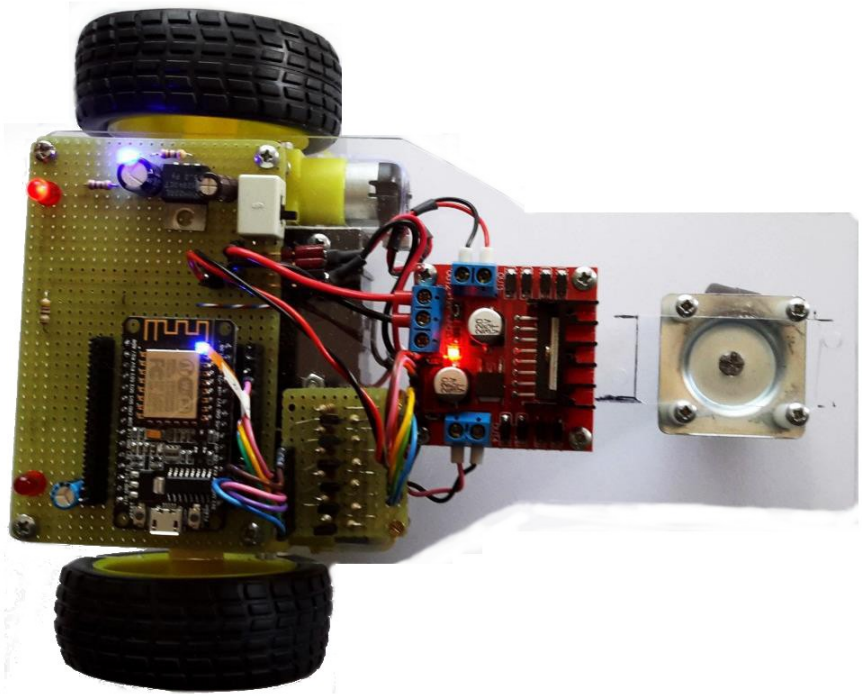

Figura 14: *Prototipo del robot.*

Indice Alfabetico

A
Access Point....8
Arduino....7, 13
B
Block....8
bootloader....5
C
comandi AT....5
E
Eclipse....7
ESP8266....5, 13
ESP8266 12E DEV KIT....5
ESP8266flasher....6
ESplorer....13
ESPlorer....6
Espressif....6
F
Flash....6
Flasher....6
Format....9
format done....9
I
IDE....7
init.lua....9
L
L298....16
level shifter....16
Log....9
Lua....5, 13
M
motor driver....16
N
nodemcu....5, 13
S
SDK....7
Send....9
sketch....14
Station....8
T
TAG....11, 17
U
USB-UART....5
W
webserver....8

Bibliografia

[1] www.LaurTec.it: sito dove scaricare l'articolo e i file allegati allo stesso.
[2] www.espressif.com : casa produttrice del transceiver ESP8266.
[3] www.nodemcu.com: Progetto nodemcu.
[4] www.youtube.com/watch?v=VvIoBFLj2Xo: video tutorial.
[5] www.github.com/nodemcu/nodemcu-devkit : Hardware
[6] www.github.com/geekscape/nodemcu_esp8266/tree/master/workshop_1
[7] Kolban's book on ESP8266 di Neil Kolban): Libro
[8] Beginning Lua programming di Kurt Jung a Aaron Brown : Libro

History

Data	Versione	Autore	Revisione	Descrizione Cambiamento
14.05.16	1.0	Paolo Salvagnini	Mauro Laurenti	Versione Originale.