**DESCRIZIONE FUNZIONALITA’**

Il progetto che abbiamo deciso di realizzare prevede la costruzione di una "macchina intelligente".

La macchina andrà a svolgere diverse funzioni, principalmente di comunicazione o di valutazione e risoluzione di problematiche, tra le quali:

* riuscire a muoversi all'interno di una strada composta da due corsie, rimanendo all'interno di una corsia
* riuscire ad effettuare una manovra di cambio corsia
* valutare la presenza di un semaforo
  + nel caso in cui venga rilevato un semaforo valuti la sua condizione e regoli la velocità di conseguenza
* valutare la presenza di possibili ostacoli durante il percorso
  + nel caso in cui venga rilevato un ostacolo valuti il tipo di ostacolo (mobile o fisso) e decida se continuare il percorso o effettuare una manovra di cambio corsia
* comunicare con un telefono per permettere lo scambio di informazioni tra la macchina e l'utente in diretta

**ELENCO COMPONENTI**

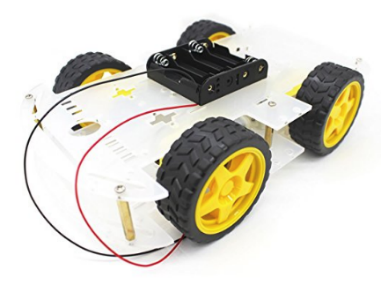
|  |  |  |
| --- | --- | --- |
| **Quantità** | **Componente** | **Link Amazon** |
| 1 | Board Arduino Mega | https://www.amazon.it/dp/B06XRJNB27/ |
| 1 | Board generica (non sono necessari molti GPIO) | https://www.amazon.it/dp/B01MRJR8UF/ |
| 1 | Kit telaio ruote motori | https://www.amazon.it/dp/B073169GBF/ |
| https://www.amazon.it/dp/B072ZWRM2B/ |
| 2 | Modulo gestione motori - **L298N** | https://www.amazon.it/dp/B013QTC18K/ |
| 2 | Modulo radio - **nRF24L01** | https://www.amazon.it/dp/B016BAPC0K/ |
| 1 | Modulo bluetooth - **HC-05** | https://www.amazon.it/dp/B01G9KSAF6/ |
| 4 | Sensore ad infrarossi - **TE174** o **FC-35** | https://www.amazon.it/dp/B00XXEJD58/ |
| 3 | Sensore ad ultrasuoni - **HC-SR04** | <https://www.amazon.it/dp/B00R2U8HK6/> |
| 1 | Resistenza x | https://www.amazon.it/ dp/B00J02WSFE/ |
| 1 | Resistenza x |
| 1 | Resistenza x AL POSTO DEI DIODI |
| 2 | Breadboard | https://www.amazon.it/ dp/B06XRDRT2R/ |
| 1 | Cavetteria | [https://www.amazon.it/ dp/B01N40EK6M/](https://www.amazon.it/%20dp/B01N40EK6M/) |

**DESCRIZIONE COMPONENTI UTILIZZATI**

Per la creazione di questo progetto sono stati utilizzati diversi componenti, aggiungendo sempre più funzionalità potendo così testare il funzionamento di diversi sensori e moduli con relative interazioni all’interno di un singolo progetto. Andremo ora ad analizzare i vari componenti:

**Telaio / Motori / Ruote**

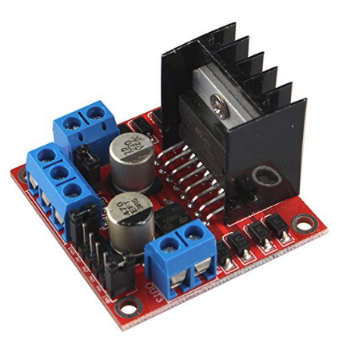
Per realizzare questo progetto è fondamentale una scocca che permetta alla macchina di sostenere i vari componenti e successivamente, una volta realizzato il codice e inseriti i vari sensori e moduli, muoversi all’interno del tracciato.

Sicuramente è un componente il cui acquisto non è necessario (sarebbe stato possibile crearlo attraverso del cartone) avendo però la necessità di comprare delle ruote e relativi motori a corrente continua la differenza di costo tra il comprare un kit telaio/motori/ruote oppure delle ruote e motori singolarmente era irrosoria se non addirittura più conveniente. Abbiamo quindi optato per questo telaio realizzato in plexiglass. I motori messi a disposizione nel kit sono dei comunissimi motori a corrente continua ai quali è possibile fornire una tensione massima di 6 Volt. Le ruote sono un mix di plastica e gomma.

Per quanto questo kit abbia comportato grandi comodità, bisogna anche dire che i motori non sono particolarmente precisi. Questo comporta la presenza di piccole imprecisioni durante la guida automatica della macchina che possono essere notate durante il suo percorso. Non dovendo però andare a realizzare un progetto che punta sulla precisione, la comodità portata dal kit unita al costo irrosorio ne fanno un acquisto decisamente sensato per il progetto.

**Modulo L298N**

Questo modulo è sicuramente un acquisto necessario quando si vuole realizzare un progetto di questo genere.

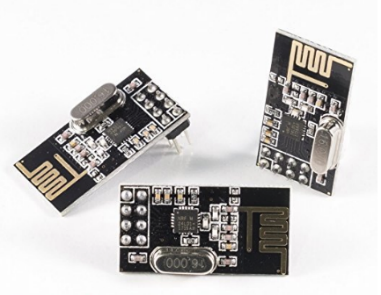


Il modulo L298N è utilizzato per comandare fino ad un massimo di due motori a corrente continua, può funzionare con tensioni che variano dai 5 ai 35 Volt e fornire sino a 2 Ampere di corrente per ogni uscita. Particolarmente utile la presenza di un’uscita stabile a 5 Volt, la quale può essere utilizzata per alimentare Arduino, dei componenti oppure per far passare 5 Volt sulla breadboard.

Per il progetto sono stati utilizzati due di questi componenti, potendo così gestire singolarmente ogni motore. Questa soluzione non è necessaria in quanto si potrebbe utilizzare un solo modulo collegando in parallelo a coppie i motori.

**Modulo nRF24L01**

Questo modulo viene utilizzato per creare un canale radio attraverso il quale comunicare. Esso non è necessario nel momento in cui si vuole costruire un semplice line follower, invece può risultare molto utile nel momento in cui si voglia comunicare con altri componenti (ad esempio un radiocomando per controllare la macchina).

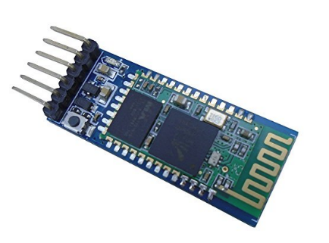


All’interno del progetto è stato inserito per permettere la comunicazione con un secondo Arduino, all’interno del quale è caricato un programma che simula un semaforo.

Il semaforo e la macchina potranno quindi comunicare tra loro, nello specifico sarà il semaforo che potrà inviare alla macchina il suo stato, la macchina andrà ad analizzare lo stato corrente e valuterà in base allo stato come comportarsi.

**Modulo HC-05**

Questo modulo non è sicuramente un acquisto necessario per il progetto, in quanto un eventuale connessione tra Arduino e un secondo Arduino è già permessa dal modulo radio.

La scelta di aggiungere un modulo la cui funzione è quella di creare una connessione Bluetooth è stata fatta per due motivi, il principale è quello di poter comunicare con un telefono tramite una delle tante applicazioni presenti sugli store e poter così effettuare un debug in live, molto utile per determinare problemi nel comportamento della macchina in fase di progettazione. In precedenza al posto del Bluetooth era presente un display LCD sul quale venivano mostrate le informazioni. Il secondo motivo è la possibilità di avere una connessione senza fili con un PC con cui scambiare dati nel caso in cui si vogliano sviluppare in futuro delle applicazioni con Processing per visualizzare il movimento della macchina.

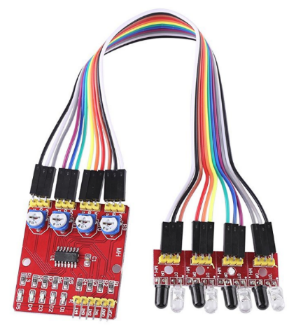
**Sensore HC-SR04**

Questo sensore permette di valutare la distanza tra esso ed un oggetto tramite l’utilizzo di ultrasuoni.

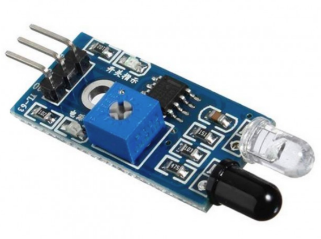


Trova un doppio utilizzo all’interno del progetto poiché risulta utile sia per la macchina che per il semaforo. Per la macchina la sua funzione è quella di permettere alla macchina di valutare la presenza di eventuali ostacoli di fronte al suo percorso, invece per quanto riguarda il semaforo il suo utilizzo è quello di creare una zona di apertura e una di chiusura di invio del segnale radio, impedendo cosi eventuali problemi dati da un segnale continuo.

**Sensore FC-35 / FC-51**



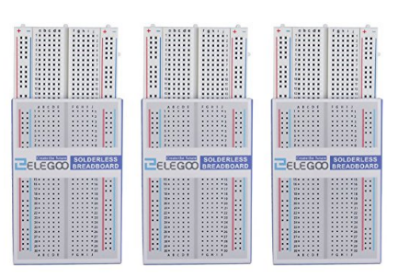
Questo modulo viene utilizzato per creare Questo modulo viene utilizzato per creare

Questo modulo viene utilizzato per creare

Questo modulo viene utilizzato per creare

Questo modulo viene utilizzato per creare

**Componentistica varia**

Per concludere, gli altri componenti utilizzati per il progetti sono una piattaforma Arduino Mega e relativa breadboard per comandare i vari moduli e sensori della macchina, un secondo Arduino Mega con breadboard per creare il semaforo.

Inoltre è stata usata molta cavetteria, dei diodi generici la cui funzione è quella di abbassare la tensione erogata dalla batteria che comanda la macchina, una resistenza da 3Kohm e una da 1Kohm, dei led per mostrare lo stato del semaforo, due pile 18650 da 3.7 Volt, due pile da 9 Volt per alimentare i due Arduino.

**CODICE REALIZZATO**

Per la realizzazione di questo progetto sono stati creati quattro file:

* **macchina.ino** per la gestione della macchina
* **semaforo.ino** per la gestione del semaforo
* **motor.h** e **motor.cpp** una libreria creata appositamente per il progetto

*/\**

*\* Nome file:* ***macchina.ino***

*\**

*\* Questo programma permette alla macchina, attraverso diverse funzioni, sensori e moduli di poter valutare*

*\* ed interagire con l’esterno. Nello specifico la macchina dovrà riuscire a muoversi all’interno di una strada*

*\* appositamente creata e comunicare attraverso segnali Bluetooth e Radio con altri oggetti.*

*\**

*\* Informazioni dettagliate al seguente link:* ***https://github.com/AndreaMagni/SistemiEmbedded/*** *\**

*\* Copyright © 2018,* ***Magni Andrea*** *&* ***Mercanti Davide***

*\**

*\* This program is free software: you can redistribute it and/or modify it under the terms of the GNU General*

*\* Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option)*

*\* any later version.*

*\**

*\* This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the*

*\* implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public*

*\* License for more details.*

*\**

*\* You should have received a copy of the GNU General Public License with this program.*

*\* If not, see <http://www.gnu.org/licenses/>.*

*\*/*

*/\**

*\* Inclusione librerie necessarie per il funzionamento del programma*

*\** ***NewPing.h*** *utilizzata per la gestione dei sensori ad ultrasuoni*

*\** ***SPI.h*** *utilizzata per la comunicazione con dispositivi SPI (serial peripheral interface)*

*\** ***nRF24L01.h*** *utilizzata per la gestione dei moduli radio*

*\** ***RF24.h*** *utilizzata per la gestione dei moduli radio*

*\** ***Motor.h*** *creata per semplificare l’utilizzo dei motori*

*\*/*

#include <NewPing.h>

#include <SPI.h>

#include <nRF24L01.h> UNA DI QUESTE DUE

#include <RF24.h> UNA DI QUESTE DUE

#include <Motor.h>

*/\**

*\* Definizione sensori ad infrarossi (nessun ostacolo rilevato [1] - ostacolo rilevato [0])*

*\** ***DETECT\_ED*** *- Sensore esterno destro*

*\** ***DETECT\_CD*** *- Sensore centrale destro*

*\** ***DETECT\_CS*** *- Sensore centrale sinistro*

*\** ***DETECT\_ES*** *- Sensore esterno sinistro*

*\*/*

#define DETECT\_ED 10

#define DETECT\_ED 11

#define DETECT\_CS 12

#define DETECT\_ES 13

*/\**

*\* Definizione variabile per effettuare il debug via bluetooth (abilitato [1] - disabilitato [0])*

*\* Utilizzare solo se strettamente necessario vista la presenza di delay nel debug*

*\* i quali possono portare ad un rallentamento delle funzioni della macchina*

*\*/*

#define DEBUG 0

*/\**

*\* Dichiarazione e assegnazione di diverse variabili. Nello specifico:*

*\** ***turn\_left****: utilizzata per*

*\** ***turn\_right****: utilizzata per*

*\** ***pipe****: utilizzata per definire l’indirizzo di comunicazione radio*

*\** ***channel****: utilizzata per definire il channel all’interno dell’indirizzo di comunicazione radio*

*\*/*

bool turn\_left = false;

bool turn\_right = false;

const uint64\_t pipe = 0xABCDEF;

const uint8\_t channel = 0x1;

*/\**

*\* Creazione oggetti*

*\** ***sonar****(trigger\_pin, echo\_pin, max\_cm\_distance)*

*\** ***mt****(speed\_[m1, m2, m3, m4], dir\_a\_[m1, m2, m3, m4], dir\_b\_[m1, m2, m3, m4])*

*\** ***radio****(ce, csn)*

*\*/*

NewPing sonar(7, 8, 200);

Motor mt(5, 3, 9, 6, 22, 26, 30, 34, 24, 28, 32, 36);

RF24 radio(49, 53);

*/\**

*\* Funzione* ***secureDigitalRead(int pin)***

*\* La funzione permette*

*\*/*

bool secureDigitalRead(int pin) {

bool register tmp1, tmp2, tmp3;

tmp1 = digitalRead(pin);

delay(2);

tmp2 = digitalRead(pin);

delay(2);

tmp3 = digitalRead(pin);

return (tmp1&&tmp2) || (tmp2&&tmp3) || (tmp1&&tmp3);

}

*/\**

*\* Funzione* ***obstacleDetection()***

*\* La funzione permette*

*\*/*

bool obstacleDetection() {

int pingTime = sonar.ping\_median(5);

int distance = sonar.convert\_cm(pingTime);

if (distance <= 15) return true;

else return false;

}

*/\**

*\* Funzione* ***segnaleSemaforo()***

*\* La funzione permette*

*\*/*

char\* segnaleSemaforo() { // DA MODIFICARE ASSOLUTAMENTE

if (radio.available()) {

char text[5] = "";

radio.read(&text, sizeof(text));

if (strcmp(text, "001") == 0) {

return ((char \*)"rosso");

} else if (strcmp(text, "010") == 0) {

return ((char \*)"giallo");

} else if (strcmp(text, "100") == 0) {

return ((char \*)"verde");

} else {

return ((char \*)"errore");

}

}

}

void setup() {

if (DEBUG) {mySerial.begin(9600);}

pinMode(DETECT\_ED, INPUT); // Definizione modalità input per il pin relativo al sensore infrarossi esterno destro

pinMode(DETECT\_CD, INPUT); // Definizione modalità input per il pin relativo al sensore infrarossi centrale destro

pinMode(DETECT\_CS, INPUT); // Definizione modalità input per il pin relativo al sensore infrarossi centrale sinistro

pinMode(DETECT\_ES, INPUT); // Definizione modalità input per il pin relativo al sensore infrarossi esterno sinistro

radio.begin();

radio.setChannel(channel);

radio.openReadingPipe(0, pipe);

radio.setPALevel(RF24\_PA\_MIN);

radio.startListening(); //INSERIRE COME CONDIZIONE

}

void loop() {

if (DEBUG) {delay(1000); mySerial.write("Rilevazione infrarossi (1 = Libero / 0 = Ostacolo)"); mySerial.write("Sensore esterno destro: "); mySerial.write(digitalRead(DETECT\_ED)); Serial.print(" - "); Serial.print("Sensore centrale destro: ");

mySerial.write(digitalRead(DETECT\_CD)); mySerial.write(" - "); mySerial.write("Sensore centrale sinistro: "); Serial.print(digitalRead(DETECT\_CS)); Serial.print(" - "); Serial.print("Sensore esterno sinistro: ");

Serial.println(digitalRead(DETECT\_ES)); Serial.println("---------------"); Serial.println("Rilevazione ultrasuono"); int pingTime = sonar.ping\_median(5); Serial.print("Ping time: "); Serial.print(pingTime); Serial.print(" sec");

Serial.print(" - "); int distance = sonar.convert\_cm(pingTime); Serial.print("Distance: "); Serial.print(distance); Serial.println(" cm"); Serial.println("---------------");}

char\* test = segnaleSemaforo();

if (strcmp(test, "rosso") == 0) {

mt.car\_stop();

if (DEBUG) {Serial.println("Rilevazione manovra in corso"); Serial.println("STOP 0"); Serial.println("---------------"); Serial.println("Rilevazione stato semaforo: "); Serial.println("rosso"); Serial.println("");}

} else {

if (obstacleDetection() && !turn\_left && secureDigitalRead(DETECT\_ED) && secureDigitalRead(DETECT\_CD) && secureDigitalRead(DETECT\_CS)) { // Verifica se la macchina puo' curvare:

mt.car\_stop();

delay(5000);

if (obstacleDetection()) {

turn\_left = true;

if (DEBUG) {mySerial.write("Rilevazione manovra in corso"); mySerial.write("Cambio corsia - Gira (da destra a sinistra)");}

mt.car\_backward(85);

delay(500);

mt.car\_left(170, 170, 170, 170);

delay(800);

if (DEBUG) {mySerial.write("Rilevazione manovra in corso"); mySerial.write("Cambio corsia - Avanti (da destra a sinistra)")}

//mt.car\_forward(70);

//delay(100);

}

} else if (obstacleDetection() && !turn\_right && secureDigitalRead(DETECT\_ES) && secureDigitalRead(DETECT\_CD) && secureDigitalRead(DETECT\_CS)) { // Verifica se la macchina puo' curvare:

mt.car\_stop();

delay(5000);

if (obstacleDetection()) {

turn\_right = true;

if (DEBUG) {lcd.clear(); lcd.print("CC-CURVA-SaD "); lcd.print(SPEED\_4); Serial.println("Rilevazione manovra in corso"); Serial.print("CAMBIO CORSIA - CURVA (da sinistra a destra) "); Serial.println(SPEED\_4); Serial.println("");}

mt.car\_backward(85);

delay(500);

mt.car\_right(170, 170, 170, 170);

delay(800);

if (DEBUG) {lcd.clear(); lcd.print("CC-AVANTI-P-SaD "); lcd.print(SPEED\_3); Serial.println("Rilevazione mnovra in corso"); Serial.print("CAMBIO CORSIA - AVANTI PRIMARIO (da sinistra a destra) "); Serial.println(SPEED\_3); Serial.println("");}

//mt.car\_forward(70);

//delay(100);

}

}

if (turn\_left) {

if (secureDigitalRead(DETECT\_CD)) {

turn\_left = false;

while (secureDigitalRead(DETECT\_CD) && !(secureDigitalRead(DETECT\_CS))) {

mt.car\_right(230, 230, 230, 230);

}

} else {

mt.car\_forward(70);

if (DEBUG) {mySerial.write("CC-AVANTI-S-DaS "); mySerial.write("CAMBIO CORSIA - AVANTI SECONDARIO (da destra a sinistra) ");}

}

} else if (turn\_right) {

if (secureDigitalRead(DETECT\_CS)) {

turn\_right = false;

while (secureDigitalRead(DETECT\_CS) && !(secureDigitalRead(DETECT\_CD))) {

mt.car\_left(230, 230, 230, 230);

}

} else {

mt.car\_forward(70);

if (DEBUG) {mySerial.write("CC-AVANTI-S-SaD "); mySerial.write("CAMBIO CORSIA - AVANTI SECONDARIO (da sinistra a destra)");}

}

} else {

if (digitalRead(DETECT\_CD) && digitalRead(DETECT\_CS)) {

if (DEBUG) {mySerial.write("Avanti");}

mt.car\_forward(75);

} else if (!(digitalRead(DETECT\_CD)) && digitalRead(DETECT\_CS)) {

if (DEBUG) {mySerial.write("Sinistra");}

mt.car\_left(130, 135, 130, 135);

} else if (digitalRead(DETECT\_CD) && !(digitalRead(DETECT\_CS))) {

if (DEBUG) {mySerial.write("Destra");}

mt.car\_right(135, 130, 135, 130);

} else if (!(digitalRead(DETECT\_CD)) && !(digitalRead(DETECT\_CS))) {

if (DEBUG) {mySerial.write("Stop");}

mt.car\_stop();

}

}

}

if (DEBUG) {mySerial.write("///////////////////////");}

}

*/\**

*\* Nome file:* ***Motor.h***

*\**

*\* Questo libreria è stata appositamente creata per poter gestier al meglio i motori vista la presenza di*

*\* un due ponti H all’interno del nostro progetto. La classe Motor presenta diversi costruttori, uno per definire*

*\* l’oggetto Motor e delle funzioni (avanti, indietro, destra, sinistra) con singolo o multiplo attributo.*

*\**

*\* Informazioni dettagliate al seguente link:* ***https://github.com/AndreaMagni/SistemiEmbedded/*** *\**

*\* Copyright © 2018,* ***Magni Andrea*** *&* ***Mercanti Davide***

*\**

*\* This program is free software: you can redistribute it and/or modify it under the terms of the GNU General*

*\* Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option)*

*\* any later version.*

*\**

*\* This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the*

*\* implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public*

*\* License for more details.*

*\**

*\* You should have received a copy of the GNU General Public License with this program.*

*\* If not, see <http://www.gnu.org/licenses/>.*

*\*/*

#ifndef Motor\_h

#define Motor\_h

#include "Arduino.h"

class Motor {

public:

Motor(int PIN\_SPEED\_M1, int PIN\_SPEED\_M2, int PIN\_SPEED\_M3, int PIN\_SPEED\_M4,

int PIN\_DIR\_A\_M1, int PIN\_DIR\_A\_M2, int PIN\_DIR\_A\_M3, int PIN\_DIR\_A\_M4,

int PIN\_DIR\_B\_M1, int PIN\_DIR\_B\_M2, int PIN\_DIR\_B\_M3, int PIN\_DIR\_B\_M4);

void car\_forward(int SPEED);

void car\_backward(int SPEED);

void car\_left(int SPEED);

void car\_right(int SPEED);

void car\_forward(int SPEED\_1, int SPEED\_2, int SPEED\_3, int SPEED\_4);

void car\_backward(int SPEED\_1, int SPEED\_2, int SPEED\_3, int SPEED\_4);

void car\_left(int SPEED\_1, int SPEED\_2, int SPEED\_3, int SPEED\_4);

void car\_right(int SPEED\_1, int SPEED\_2, int SPEED\_3, int SPEED\_4);

void car\_stop();

private:

int \_PIN\_SPEED\_M1;

int \_PIN\_SPEED\_M2;

int \_PIN\_SPEED\_M3;

int \_PIN\_SPEED\_M4;

int \_PIN\_DIR\_A\_M1;

int \_PIN\_DIR\_A\_M2;

int \_PIN\_DIR\_A\_M3;

int \_PIN\_DIR\_A\_M4;

int \_PIN\_DIR\_B\_M1;

int \_PIN\_DIR\_B\_M2;

int \_PIN\_DIR\_B\_M3;

int \_PIN\_DIR\_B\_M4;

};

#endif

*/\**

*\* Nome file:* ***Motor.ccp***

*\**

*\* Questo libreria è stata appositamente creata per poter gestier al meglio i motori vista la presenza di*

*\* un due ponti H all’interno del nostro progetto. La classe Motor presenta diversi costruttori, uno per definire*

*\* l’oggetto Motor e delle funzioni (avanti, indietro, destra, sinistra) con singolo o multiplo attributo.*

*\**

*\* Informazioni dettagliate al seguente link:* ***https://github.com/AndreaMagni/SistemiEmbedded/*** *\**

*\* Copyright © 2018,* ***Magni Andrea*** *&* ***Mercanti Davide***

*\**

*\* This program is free software: you can redistribute it and/or modify it under the terms of the GNU General*

*\* Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option)*

*\* any later version.*

*\**

*\* This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the*

*\* implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public*

*\* License for more details.*

*\**

*\* You should have received a copy of the GNU General Public License with this program.*

*\* If not, see <http://www.gnu.org/licenses/>.*

*\*/*

#include "Arduino.h"

#include "Motor.h"

Motor::Motor(int PIN\_SPEED\_M1, int PIN\_SPEED\_M2, int PIN\_SPEED\_M3, int PIN\_SPEED\_M4, int PIN\_DIR\_A\_M1,

int PIN\_DIR\_A\_M2, int PIN\_DIR\_A\_M3, int PIN\_DIR\_A\_M4, int PIN\_DIR\_B\_M1, int PIN\_DIR\_B\_M2,

int PIN\_DIR\_B\_M3, int PIN\_DIR\_B\_M4) {

pinMode(PIN\_SPEED\_M1, OUTPUT);

pinMode(PIN\_DIR\_A\_M1, OUTPUT);

pinMode(PIN\_DIR\_B\_M1, OUTPUT);

pinMode(PIN\_SPEED\_M2, OUTPUT);

pinMode(PIN\_DIR\_A\_M2, OUTPUT);

pinMode(PIN\_DIR\_B\_M2, OUTPUT);

pinMode(PIN\_SPEED\_M3, OUTPUT);

pinMode(PIN\_DIR\_A\_M3, OUTPUT);

pinMode(PIN\_DIR\_B\_M3, OUTPUT);

pinMode(PIN\_SPEED\_M4, OUTPUT);

pinMode(PIN\_DIR\_A\_M4, OUTPUT);

pinMode(PIN\_DIR\_B\_M4, OUTPUT);

\_PIN\_SPEED\_M1 = PIN\_SPEED\_M1;

\_PIN\_DIR\_A\_M1 = PIN\_DIR\_A\_M1;

\_PIN\_DIR\_B\_M1 = PIN\_DIR\_B\_M1;

\_PIN\_SPEED\_M2 = PIN\_SPEED\_M2;

\_PIN\_DIR\_A\_M2 = PIN\_DIR\_A\_M2;

\_PIN\_DIR\_B\_M2 = PIN\_DIR\_B\_M2;

\_PIN\_SPEED\_M3 = PIN\_SPEED\_M3;

\_PIN\_DIR\_A\_M3 = PIN\_DIR\_A\_M3;

\_PIN\_DIR\_B\_M3 = PIN\_DIR\_B\_M3;

\_PIN\_SPEED\_M4 = PIN\_SPEED\_M4;

\_PIN\_DIR\_A\_M4 = PIN\_DIR\_A\_M4;

\_PIN\_DIR\_B\_M4 = PIN\_DIR\_B\_M4;

}

void Motor::car\_forward(int SPEED) {

analogWrite(\_PIN\_SPEED\_M1, SPEED);

analogWrite(\_PIN\_SPEED\_M2, SPEED);

analogWrite(\_PIN\_SPEED\_M3, SPEED);

analogWrite(\_PIN\_SPEED\_M4, SPEED);

digitalWrite(\_PIN\_DIR\_A\_M1, LOW);

digitalWrite(\_PIN\_DIR\_B\_M1, HIGH);

digitalWrite(\_PIN\_DIR\_A\_M2, LOW);

digitalWrite(\_PIN\_DIR\_B\_M2, HIGH);

digitalWrite(\_PIN\_DIR\_A\_M3, LOW);

digitalWrite(\_PIN\_DIR\_B\_M3, HIGH);

digitalWrite(\_PIN\_DIR\_A\_M4, LOW);

digitalWrite(\_PIN\_DIR\_B\_M4, HIGH);

}

void Motor::car\_forward(int SPEED\_1, int SPEED\_2, int SPEED\_3, int SPEED\_4) {

analogWrite(\_PIN\_SPEED\_M1, SPEED\_1);

analogWrite(\_PIN\_SPEED\_M2, SPEED\_2);

analogWrite(\_PIN\_SPEED\_M3, SPEED\_3);

analogWrite(\_PIN\_SPEED\_M4, SPEED\_4);

digitalWrite(\_PIN\_DIR\_A\_M1, LOW);

digitalWrite(\_PIN\_DIR\_B\_M1, HIGH);

digitalWrite(\_PIN\_DIR\_A\_M2, LOW);

digitalWrite(\_PIN\_DIR\_B\_M2, HIGH);

digitalWrite(\_PIN\_DIR\_A\_M3, LOW);

digitalWrite(\_PIN\_DIR\_B\_M3, HIGH);

digitalWrite(\_PIN\_DIR\_A\_M4, LOW);

digitalWrite(\_PIN\_DIR\_B\_M4, HIGH);

}

void Motor::car\_backward(int SPEED) {

analogWrite(\_PIN\_SPEED\_M1, SPEED);

analogWrite(\_PIN\_SPEED\_M2, SPEED);

analogWrite(\_PIN\_SPEED\_M3, SPEED);

analogWrite(\_PIN\_SPEED\_M4, SPEED);

digitalWrite(\_PIN\_DIR\_A\_M1, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M1, LOW);

digitalWrite(\_PIN\_DIR\_A\_M2, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M2, LOW);

digitalWrite(\_PIN\_DIR\_A\_M3, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M3, LOW);

digitalWrite(\_PIN\_DIR\_A\_M4, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M4, LOW);

}

void Motor::car\_backward(int SPEED\_1, int SPEED\_2, int SPEED\_3, int SPEED\_4) {

analogWrite(\_PIN\_SPEED\_M1, SPEED\_1);

analogWrite(\_PIN\_SPEED\_M2, SPEED\_2);

analogWrite(\_PIN\_SPEED\_M3, SPEED\_3);

analogWrite(\_PIN\_SPEED\_M4, SPEED\_4);

digitalWrite(\_PIN\_DIR\_A\_M1, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M1, LOW);

digitalWrite(\_PIN\_DIR\_A\_M2, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M2, LOW);

digitalWrite(\_PIN\_DIR\_A\_M3, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M3, LOW);

digitalWrite(\_PIN\_DIR\_A\_M4, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M4, LOW);

}

void Motor::car\_left(int SPEED) {

analogWrite(\_PIN\_SPEED\_M1, SPEED);

analogWrite(\_PIN\_SPEED\_M2, SPEED);

analogWrite(\_PIN\_SPEED\_M3, SPEED);

analogWrite(\_PIN\_SPEED\_M4, SPEED);

digitalWrite(\_PIN\_DIR\_A\_M1, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M1, LOW);

digitalWrite(\_PIN\_DIR\_A\_M2, LOW);

digitalWrite(\_PIN\_DIR\_B\_M2, HIGH);

digitalWrite(\_PIN\_DIR\_A\_M3, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M3, LOW);

digitalWrite(\_PIN\_DIR\_A\_M4, LOW);

digitalWrite(\_PIN\_DIR\_B\_M4, HIGH);

}

void Motor::car\_left(int SPEED\_1, int SPEED\_2, int SPEED\_3, int SPEED\_4) {

analogWrite(\_PIN\_SPEED\_M1, SPEED\_1);

analogWrite(\_PIN\_SPEED\_M2, SPEED\_2);

analogWrite(\_PIN\_SPEED\_M3, SPEED\_3);

analogWrite(\_PIN\_SPEED\_M4, SPEED\_4);

digitalWrite(\_PIN\_DIR\_A\_M1, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M1, LOW);

digitalWrite(\_PIN\_DIR\_A\_M2, LOW);

digitalWrite(\_PIN\_DIR\_B\_M2, HIGH);

digitalWrite(\_PIN\_DIR\_A\_M3, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M3, LOW);

digitalWrite(\_PIN\_DIR\_A\_M4, LOW);

digitalWrite(\_PIN\_DIR\_B\_M4, HIGH);

}

void Motor::car\_right(int SPEED) {

analogWrite(\_PIN\_SPEED\_M1, SPEED);

analogWrite(\_PIN\_SPEED\_M2, SPEED);

analogWrite(\_PIN\_SPEED\_M3, SPEED);

analogWrite(\_PIN\_SPEED\_M4, SPEED);

digitalWrite(\_PIN\_DIR\_A\_M1, LOW);

digitalWrite(\_PIN\_DIR\_B\_M1, HIGH);

digitalWrite(\_PIN\_DIR\_A\_M2, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M2, LOW);

digitalWrite(\_PIN\_DIR\_A\_M3, LOW);

digitalWrite(\_PIN\_DIR\_B\_M3, HIGH);

digitalWrite(\_PIN\_DIR\_A\_M4, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M4, LOW);

}

void Motor::car\_right(int SPEED\_1, int SPEED\_2, int SPEED\_3, int SPEED\_4) {

analogWrite(\_PIN\_SPEED\_M1, SPEED\_1);

analogWrite(\_PIN\_SPEED\_M2, SPEED\_2);

analogWrite(\_PIN\_SPEED\_M3, SPEED\_3);

analogWrite(\_PIN\_SPEED\_M4, SPEED\_4);

digitalWrite(\_PIN\_DIR\_A\_M1, LOW);

digitalWrite(\_PIN\_DIR\_B\_M1, HIGH);

digitalWrite(\_PIN\_DIR\_A\_M2, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M2, LOW);

digitalWrite(\_PIN\_DIR\_A\_M3, LOW);

digitalWrite(\_PIN\_DIR\_B\_M3, HIGH);

digitalWrite(\_PIN\_DIR\_A\_M4, HIGH);

digitalWrite(\_PIN\_DIR\_B\_M4, LOW);

}

void Motor::car\_stop() {

analogWrite(\_PIN\_SPEED\_M1, 0);

analogWrite(\_PIN\_SPEED\_M2, 0);

analogWrite(\_PIN\_SPEED\_M3, 0);

analogWrite(\_PIN\_SPEED\_M4, 0);

digitalWrite(\_PIN\_DIR\_A\_M1, LOW);

digitalWrite(\_PIN\_DIR\_B\_M1, LOW);

digitalWrite(\_PIN\_DIR\_A\_M2, LOW);

digitalWrite(\_PIN\_DIR\_B\_M2, LOW);

digitalWrite(\_PIN\_DIR\_A\_M3, LOW);

digitalWrite(\_PIN\_DIR\_B\_M3, LOW);

digitalWrite(\_PIN\_DIR\_A\_M4, LOW);

digitalWrite(\_PIN\_DIR\_B\_M4, LOW);

}