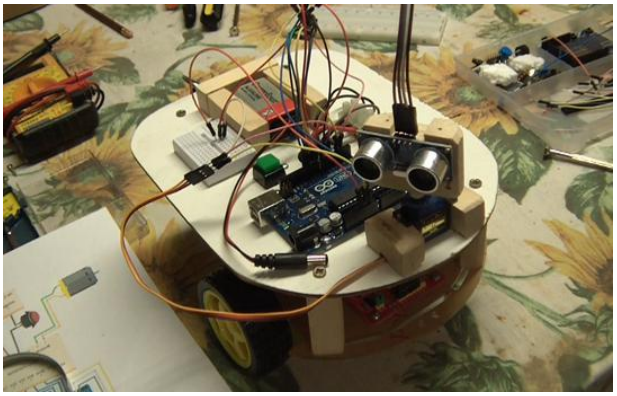
A – Poliphemus – carrello robot con sensore ad ultrasuoni - Cart driven by an ultrasound sensor (some notes at end of this section)



Poliphemus e' un carrello mosso da due motori pilotati dalla scheda L298, alimentata da una batteria da 6 volt e 4,2 ah. Il carrello e' dotato di un sensore ad ultrasuoni che gli consente di rilevare gli ostacoli e di un servomotore che fa ruotare di 180 gradi il sensore, in modo da consentirgli anche di identificare gli ostacoli laterali. Il programma unisce le esperienze accumulate con gli esercizi 20, 21 e 22 (che sarebbe opportuno replicare o comunque leggere, prima di cimentarsi nella costruzione del carrello) mentre i componenti sono reperibili a basso costo sui mercati online cinesi (i componenti, scheda arduino e carrello compresi, utilizzati nel prototipo e nei primi 22 esercizi di questa raccolta sono costati in tutto meno di 50 euro). La batteria ed il caricabatterie invece, acquistati in un magazzino materiale elettrico situato in Italia, sono costati quasi come tutti gli altri componenti messi insieme.

In aggiunta a quanto sopra la costruzione del carrello ha richiesto un po' di impegno nel taglio, sagomatura e montaggio del piano sopraelevato, sul quale sono stati fissati il servomotore, il sensore ad ultrasuoni e la scheda arduino. Il filmato, reperibile qui', mostra seppur in maniera velocizzata, le fasi ed i particolari di costruzione.

La gestione dei motori a spazzola, pur magnificamente supportata dalla scheda L298N, e' resa problematica dall'impossibilita' di controllarne compiutamente velocita' e numero di giri di ogni motore per cui il prototipo, pur interessante in se', sembra difficilmente migliorabile in termini di precisione e linearita' di percorso.

Nota: Questo prototipo e questa nota sono parte di una serie che vede protagonisti Arduino ed alcuni dei componenti ad esso collegabili. Per la maggior parte degli esercizi e' anche disponibile un filmato su youtube.

- Esercizi facenti parte della raccolta
- Filmati presenti su youtube
- Informazioni su arduino e sui componenti collegabili (PDF scaricato nell'area di download)
- Breve manuale di programmazione (PDF scaricato nell'area di download)

Per eventuali chiarimenti o suggerimenti sul contenuto di questa scheda scrivere a giocarduino@libero.it

Here some notes about this project, translated by google translator

Polyphemus is a cart managed by two motors driven by a L298 device and powered by a 6-volt battery and 4.2 ha. The cart is equipped with an ultrasound sensor which allows it to detect obstacles and a servo motor that rotates the sensor so as to enable also the identification of side obstacles. The program combines the experience done in examples 20, 21 and 22 (which should be replicated or at least read, before engage in the trolley construction).

The construction require a little commitment in cutting, shaping and assembly of the raised floor, on which were set servo motor, ultrasound sensor and Arduino. The video, available here, shows although sped up, phases and construction details.

The brush motors management, magnificently supported by L298N device, is made problematic by the impossibility to fully control the pivot speed. For that reason the prototype, although interesting in itself, it seems difficult to improve in terms of precision & linearity path.

Note: This project and this note is part of a series that sees, as main characters, Arduino and some of connectable components. For most projects there is also a video on youtube.

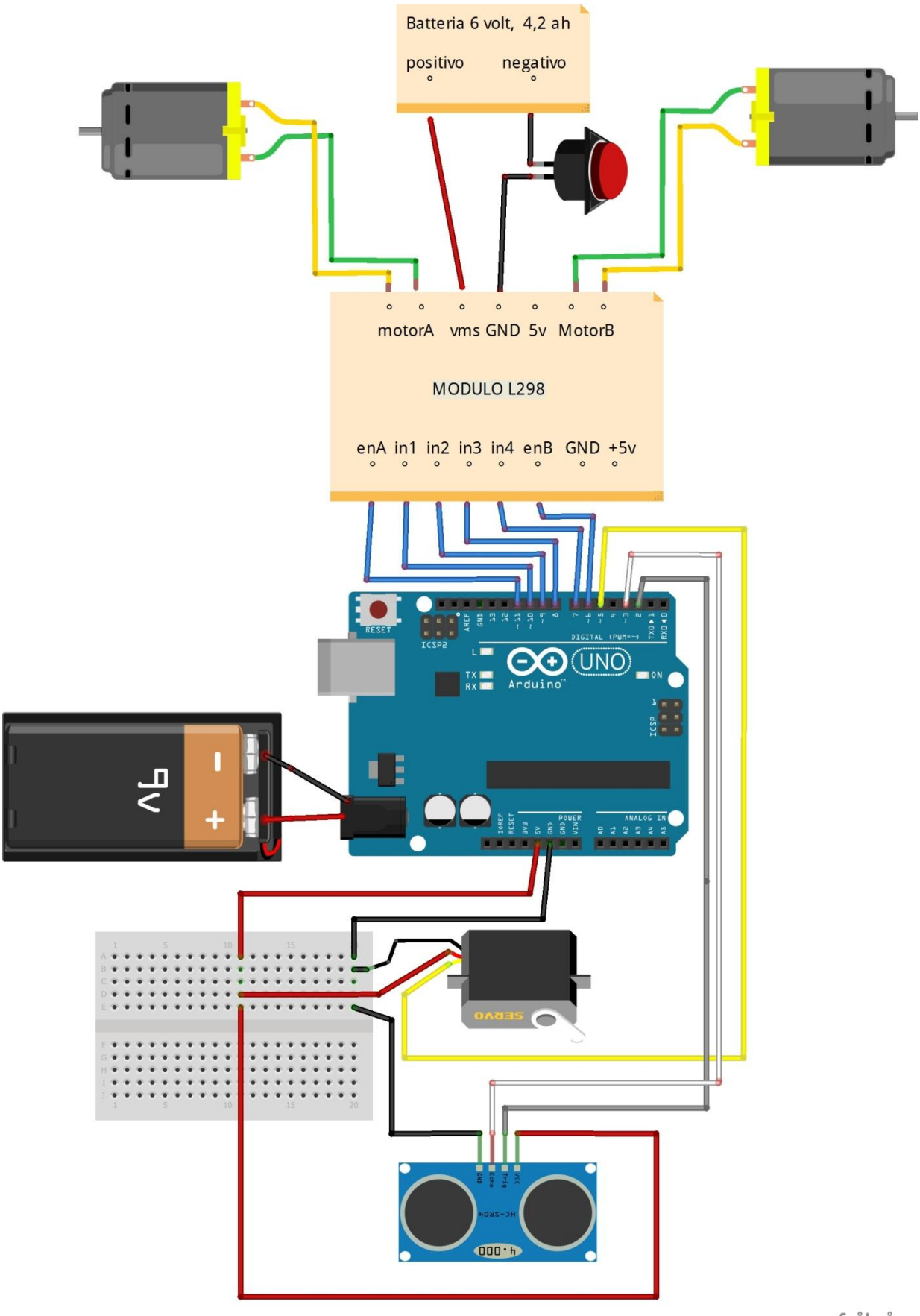
- Projects collection
- Movies on youtube
- About Arduino and components (italian; pdf will be downloaded in your download area
- Quick programming guide (almost english; pdf will be downloaded in your download area)

For any questions or suggestions about this note (and on its english translation), please write to giocarduino@libero.it (simple words and short sentences, please)

Materiali

- 1 scheda Arduino
- 1 scheda L298N
- 1 servomotore microservo 9g (sg90)
- 1 modulo ad ultrasuoni HC-SR04
- 1 batteria da 6 volt per l'alimentazione dei motori (sono possibili anche voltaggi maggiori, anche fino a 20 volt, a patto che i motori siano in grado di reggerli)
- 1 batteria da 9 volt per l'alimentazione di Arduino
- 1 interruttore on/off (non indispensabile)
- 1 carrellino (base + motori e ruote) autocostruito o reperibile sui mercati orientali
- Un po' di cavetteria

Schema


Batteria 6 volt, 4,2 ah
positivo
negativo
motorA vms GND 5v MotorB
MODULO L298
enA in1 in2 in3 in4 enB GND +5v
RESET
ICSP2
UNO
Arduino
TX
RX
ON
ICSP
DIGITAL (PWM~)
POWER
ANALOG IN
9V
SERVO
fritzing

Programma

```
/*  Attenzione: facendo il copia/incolla dal PDF all'IDE si perde la formattazione del testo.
Per rendere piu' facilmente leggibile il programma e' opportuno formattarlo subito dopo il
trasferimento nell'IDE, premendo CTRL+T. Poliphemus e' un carrello mosso da due motori a spazzole
pilotati dalla scheda L298. Ha un solo occhio, un sensore ad ultrasuoni attraverso il quale
individua gli ostacoli e li evita. Il sensore e' montato su di un servomotore Sg90 che lo fa ruotare
di 180 gradi e che gli consente quindi di individuare eventuali ostacoli  posti ai lati.
*
Poiche' e' quasi impossibile reperire due motori che, a parita' di tensione, offrono le medesime
performance, nella routine di gestione dei motori e' stato previsto un "valore di gap" applicato ad
uno dei due motori che deve essere sperimentalmente definito (puo' essere sia positivo che negativo)
per cercare di avvicinare le prestazioni dei due motori.
*
Sia il valore di gap che la configurazione delle porte In1-In4 della scheda L298 possono cambiare in
funzione delle modalita' di cablaggio e del divario di prestazioni dei motori. Un'eventuale replica
di questo prototipo deve essere quindi sperimentalmente "aggiustata"
*
*-----------------------------------------------------------------------------------------------
* Warning: cut&paste from PDF to IDE loses formatting. to restore it press CTRL + T.
Poliphemus is a cart driven by two brush motors driven by the L298 card. It has only one eye, an
ultrasonic sensor through which identifies obstacles. The sensor is mounted on a SG90 servo motor
that rotates 180 degrees and that then allows it to identify any obstacles to the sides.
*
Since is almost impossible find two engines that provide the same performance, in the engine
management routines there is a "gap corrector" value. This value, applied to one of the two engines
tries to approach the performance of the two motors. Value of "gap corrector" must be experimentally
defined (It can be both positive and negative)
*
the value of "gap corrector" and the configuration of the In1-In4 pins in L298 device, may change
depending on the  wiring mode and motors performance.
*
*-----------------------------------------------------------------------------------------------
*/

// variabili per la gestione del servomotore - variabiles to manage servo
//
#include <Servo.h> // richiama la libreria di gestione del servomotore
Servo pippo;       // crea il servo oggetto "pippo" da utilizzare per riferire il servomotore
int pos = 0;       // posizione, in gradi angolari, da far raggiungere al perno del servomotore
int posprec = 0;   // posizione corrente del perno del servomotore
int gradi = 0;     // posizione di partenza del servomotore
int incremento = 1;// step di incremento o decremento dei gradi da far compiere al servomotore
//
//
//  variabili per la gestione dei motori del carrello - variabile to manage engines
//
int ENA=11;        // motore A (dx) connesso alla porta 11 (output pwm)
int IN1=10;        //connesso alla porta 10
int IN2=9;         // connesso alla porta 9
int IN3=8;         // connesso alla porta 8
int IN4=7;         // connesso alla porta 7
int ENB=6;         // motore B (sx)connesso alla porta 6 (output pwm)
//
int gap = 0;       /* correttore di gap - gap corrector value
Valore utilizzato nell'istruzione AnalogWrite per compensare il gap prestazionale di un motore
rispetto all'altro. Agisce sul voltaggio del motore A e puo' essere  sia positivo (aumenta la
tensione e quindi la velocita' di rotazione) oppure negativo (diminuisce la tensione e quindi
la velocita' di rotazione - value used in analogWrite statement to compensate performance gap
of a motor relative to each other. It acts on the voltage of the motor A and can be both
positive (increase the voltage and therefore the speed of rotation) or negative (decrease the
voltage and thus the speed of rotation) */
//
int velocita = 255;// parametro "velocita" dei motori (min 120 max 255, gap compreso) - "speed"
//parameter for engines (min 120 max 255, including gap)
//
int rotazione90 = 500; // tempo (in millisecondi) per una rotazione di 90 gradi del carrello - time
//(in milliseconds) for a 90 degree rotation of the carriage
//
// variabili per la gestione del sensore ad ultrasuoni - variabiles to manage ultrasound sensor
//
float cm;          // variabile in cui viene inserita la distanza dall'ostacolo, in centimetri
//
// ****************** routine di avanzamento del carrello ******************************
//*******************************  foward  ********************************************
//
void avanti (void)
```

```
{
  digitalWrite(IN1,0);
  digitalWrite(IN2,1);
  digitalWrite(IN3,1);
  digitalWrite(IN4,0);
  analogWrite(ENA, velocita + gap);     //  Attiva il motore A
  analogWrite(ENB, velocita  );         //  Attiva il motore B
}
//
// ********* routine rotazione del carrello a sx di 90 gradi ***************
// ********************** 90 left degree ********************************
//
void sx90 (void)                {
  digitalWrite(IN1,1);
  digitalWrite(IN2,0);
  digitalWrite(IN3,1);
  digitalWrite(IN4,0);
  analogWrite(ENA, velocita + gap);
  analogWrite(ENB, velocita );
  delay (rotazione90);
  analogWrite(ENA,0);                //  blocca il motore A
  analogWrite(ENB,0);                //  blocca il motore B
}
//
// *********** routine rotazione del carrello a dx di 90 gradi **************
// ********************** 90 rigth degree ********************************
//
void dx90 (void)
{
  digitalWrite(IN1,0);
  digitalWrite(IN2,1);
  digitalWrite(IN3,0);
  digitalWrite(IN4,1);
  analogWrite(ENA, velocita + gap);
  analogWrite(ENB, velocita );
  delay (rotazione90);
  analogWrite(ENA,0);
  analogWrite(ENB,0);
}
//
//  *************** routine di verifica ostacoli ********************************
//  *********************obstacle check ***********************************************
//
void sensore (void)
{
  gradi = posprec;
  incremento = 1;
  if (pos <= posprec)
    incremento = -1;
  for(posprec = gradi; posprec != pos; posprec = posprec + incremento)  // sposta di un grado
 //  per volta, da posprec a pos, l'angolazione desiderata
  {
    pippo.write(posprec); // indirizza il perno alla posizione desiderata, memorizzata in 'posprec'
    delay(15);            // attende 15ms per consentire al servomotore di raggiungere la posizione
  }
  digitalWrite(2, LOW);   //disattiva il lancio del fascio di ultrasuoni (qualora fosse attivo)
  delayMicroseconds(2);   // attende 2 microsecondi
  digitalWrite(2, HIGH);  // attiva il lancio del fascio di ultrasuoni
  delayMicroseconds(10);  // attende 10 microsecondi (il tempo richiesto dal modulo HC-SR04)
  digitalWrite(2, LOW);   // disattiva il lancio del fascio di ultrasuoni
  cm = pulseIn(3, HIGH) / 58.0;   // rileva il segnale di ritorno e lo converte in centimetri
}
//
//
void setup()
{
  delay (3000);         // attende 3 secondi prima di avviare il carrello
  pippo.attach(5);      // assegna il servo oggetto "pippo" alla porta 5
  pippo.write (90);     // posiziona il sensore in linea con il senso di marcia
  posprec = 90;         // memorizza la posizione del sensore
  pinMode(ENA,OUTPUT); // definisce come output tutte  le porte di gestione dei motori
  pinMode(ENB,OUTPUT);
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);
  pinMode(2, OUTPUT);  // definisce la porta digitale 2 (il trigger del sensore ad ultrasuoni)
                       // come porta di output
```

```
  pinMode(3, INPUT);    // definisce la porta digitale 3 (l'echo del sensore ad ultrasuoni)
                        // come porta di input
  analogWrite(ENA,0);   // blocca il motore A
  analogWrite(ENB,0);   // blocca il motore B
}
//
//
void loop()
{
  pos = 90;             // posiziona il sensore in linea con il senso di marcia
  sensore ();           // Verifica la presenza di un ostacolo davanti al senso di marcia
  if (cm > 30)
  {
    avanti();           // se l'ostacolo e' lontano attiva e mantiene attivo il carrello
  }
  else                  // se l'ostacolo e' vicino
  {
    analogWrite(ENA,0);  //  blocca il motore A
    analogWrite(ENB,0);  //  blocca il motore B
    pos = 0;           // verifica la presenza di ostacoli a dx
    sensore ();
    if (cm > 30)       // se non ci sono ostacoli a dx
    {
      dx90 ();         // gira il carrello a dx di 90 gradi
    }
    else               // se ci sono ostacoli a dx
    {
      pos = 180;       // verifica la presenza di ostacoli a sinistra
      sensore ();
      if (cm > 30)     // se non ci sono ostacoli a sinistra
      {
        sx90 ();     // gira il carrello a sx di 90 gradi
      }
      else           // se ci sono ostacoli sia davanti, che a dx che a sx
      {
        sx90 ();     // inverte il senso di marcia (due rotazioni a sx di 90 gradi)
        sx90 ();
      }
    }
  }
}
```