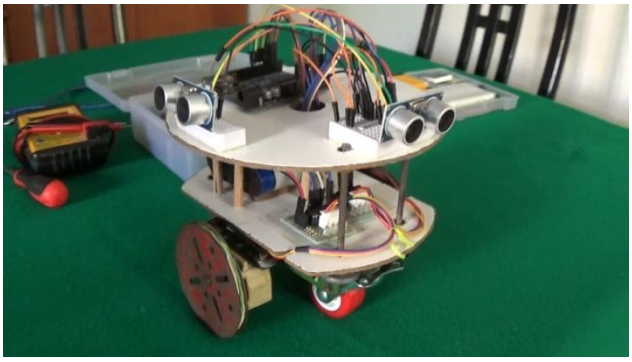
T- Thesus: fuga dal labirinto – maze escape (some notes at section end)



Theseus e' un carrello mosso da due motori a quattro poli che, sulla base delle informazioni provenienti da due sensori ad ultrasuoni in grado di "vedere" gli ostacoli davanti ed a destra, si muove all'interno di un labirinto e percorre ogni possibile strada sino a quando non trova l'uscita. Il carrello e' di forma circolare ed ha un diametro di poco meno di 20 cm mentre il labirinto, per poter essere percorso, deve essere composto da corridoi larghi almeno 30 cm e deve avere le deviazioni ad angolo retto. Il labirinto, inoltre, non deve prevedere percorsi circolari poiche' in questo caso Theseus entrerebbe in loop e continuerebbe a percorre la medesima strada sino a quando le batterie saranno in grado di muovere i motori.

Il moto del carrello e' basato su tre regole:

1. se non ci sono ostacoli o aperture sulla destra, il carrello prosegue diritto
2. se c'e' un'apertura sulla destra il carrello gira a destra e si infila nel varco
3. se c'e' un ostacolo (se si trova in un vicolo cieco) il carrello fa dietrofront

Il carrello e' anche dotato di un sistema di valutazione della posizione rispetto alla parete destra del corridoio, dalla quale cerca di restare ad una distanza compresa tra i 5 e gli 8 cm. Nel momento in cui si accorge che la distanza e' maggiore o minore, corregge la sua traiettoria per rientrare nei parametri previsti.

Nella costruzione del carrello sono stati riscontrati parecchi problemi di tipo meccanico, dovuti essenzialmente al diverso "gioco" presente negli apparati riduttori dei due motori. Per risolvere il problema e' stato introdotto il parametro "**correttore**" che aumenta (o diminuisce, a seconda dei casi) gli impulsi inviati ad un motore, in modo da compensare il gioco nel momento in cui una ruota inverte il senso rotazione.

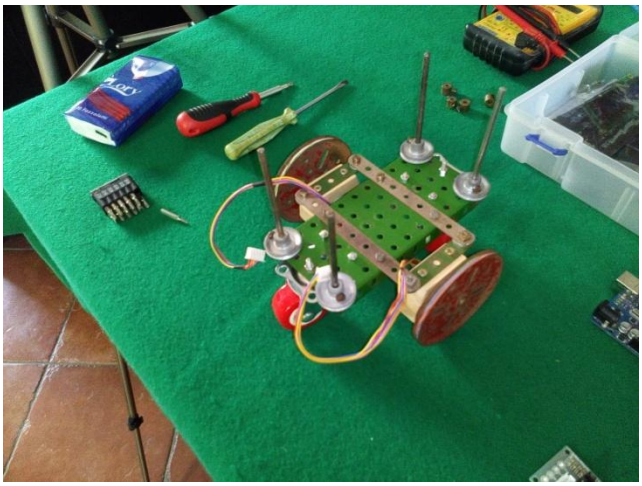
Questo e' un progetto che unisce un una certa complessita' meccanica ad una certa complessita' di programmazione per cui, qualora lo si volesse replicare, sara' necessario comprendere la funzione dei vari parametri ed agire su di essi al fine di adattarli alle dimensioni ed alla meccanica del carrello che si intende utilizzare. Qui il filmato del progetto.

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

- Progetti 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



Theseus is a cart driven by two stepper motors that, based on information from two ultrasonic sensors, moves inside a maze and runs every possible road as long as it doesn't find the exit.

The cart is circular and has a diameter of less than 20 cm while the maze must be made up of corridors of at least 30 cm wide and must have 90 degree angular deviations.

The maze, moreover, doesn't have circular paths, because in this case Theseus would enter the loop and will continue travel the loop until the batteries will be able to move motors.

Cart moves observing three rules:

1. If there are no obstructions or openings on the right, it goes straight
2. If there is an opening on the right, it runs to the right
3. If there is an obstacle it turns and goes back

Cart is also equipped with a positioning system with respect the right wall of the corridor, from which it tries to stay within a distance from 5 to 8 cm. When it notes that distance is bigger or shorter, corrects its trajectory to fit in the predicted parameters.

Some mechanical problems were found in the cart construction, due essentially to the different backlash on the engines gearboxes. To solve problem, the "**correttore**" parameter has been introduced, which increases or decreases pulses sent to a motor, to compensate the backlash when a wheel inverts rotation.

This project combines a certain mechanical complexity with a certain programming complexity. If you want to replicate it, you will need to understand the function of each parameters and act in order to adapt them to dimensions and to mechanics in cart that you are going to use. Here the project movie

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)

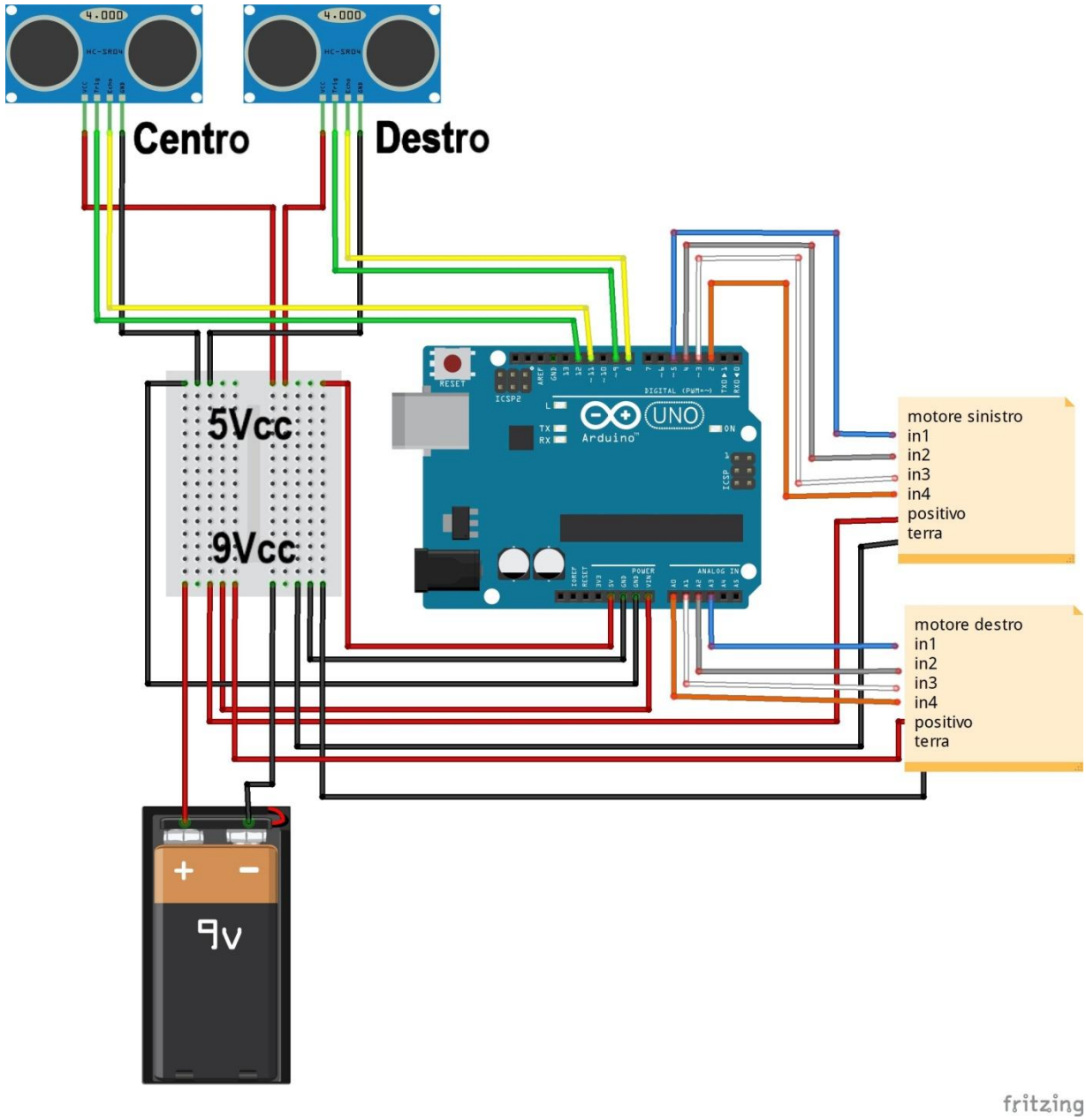
For any questions or suggestions about this note (and on its english translation), please write to giocarduino@libero.it (simple words and short sentences, please)

Materiali

- Due sensori ad ultrasuoni di tipo HC-SR04
- Due motori a 4 poli (due stepper motors) di tipo 28YBJ-48, con i relativi driver
- Una batteria da 9 volt
- Alcuni pezzi di meccano

Arduino: fuga dal labirinto – maze escape

Schema



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.

 Theseus e' un carrello mosso da due motori a 4 poli. La sua specialita' e' trovare la strada per
 uscire da un labirinto. Il carrello "vede" il labirinto attraverso due sensori ad ultrasuoni del
 tipo HC-SR04, piazzati davanti e sul lato destro. Il carrello prende sempre la prima via a destra
 ed e'in grado di tornare sui suoi passi e battere una nuova strada nel momento in cui si accorge
 di essere entrato in una strada cieca.  Non e' in grado di uscire da un percorso circolare, per
 cui nel disegnare il labirinto bisogna evitare la presenza di detti percorsi

  Sotto l'aspetto tecnico il carrello percorre circa 4 cm, dopodiche' si ferma per verificare
  la distanza dalla parete dx e, se del caso, modificare leggermente la traiettoria in modo da
  portarsi al centro del corridoio.

  il programma prevede un "correttore" per compensare l'inerzia meccanica di un motore
  rispetto all'altro. Per inerzia meccanica si intende la differenza di gioco tra i perni
  in uscita dai due riduttori, gioco dovuto a millesimali differenze nella dentatura e
  nella distanza tra gli ingranaggi dei due apparati motore
```

Arduino: fuga dal labirinto – maze escape

```
  Nel carrello utilizzato nel filmato che illustra questo progetto, il motore di destra
  ha una inerzia meccanica maggiore di circa 45 impulsi rispetto al motore di sinistra

  Qualche grandezza fisica :
  Circonferenza del carrello                    175 mm
  Ampiezza del corridoio                        300 mm
  impulsi per rotazione completa di ogni ruota: 2048
  distanza tra le ruote                         145 mm
  diametro ruote motrici:                        75 mm
  da questi ultime tre grandezze consegue che:
  circonferenza ruote motrici                   235 mm
  avanzamento lineare ad ogni impulso:            0,1147 mm
  numero di impulsi per percorrere 4 cm:        348
  impulsi sui due motori per rotazione 90 gradi: 992

  Le grandezze fisiche di cui sopra variano al variare delle dimensioni delle ruote ed al variare
  della distanza tra le ruote motrici.
```

```
  Warning: cut&paste from PDF to IDE loses formatting. To restore it press CTRL + T.

  Theseus is a cart driven by two 4-pole motors. His specialty is find the way out from a maze. The
  cart "sees" the maze through two ultrasound sensors HC-SR04 type, placed in front and on the right
  side. Cart always takes the first road to the right  and  is able to go back on his steps and run a
  new road as it realizes to have entered a blind road. He can't get out of a circular path so, in
  drawing the maze, the presence of these paths must be avoided.

  From a technical point of view, the cart runs about 4 cm, after which it stops and check distance
  from wall on the right and, where appropriate, slightly modify trajectory to take the center of the
  corridor.

  The program includes a corrector "correttore" to compensate the mechanical inertia of an engine
  respect to the other. For "mechanical inertia" we intend the different backlash between the stepper
  motors pins, backlash due to micrometeric differences in gears tooths and axis distance, in the two
  gearboxes

  In cart used to test this program ( and in the movie that illustrates this project), the right
  engine has a mechanical inertia of about 45 pulses greater respect to the left engine

   some phisical dimension:
   Cart circumference                           175 mm
   corridor amplitude:                          300 mm
   pulses for a complete wheel rotation:       2048
   wheel distance                               145 mm
   Wheel diameter:                               75 mm
   From these last three sizes it follows that:
   wheel circumference:                         235 mm
   Linear advancement at each   pulse:            0,1147 mm
   pulses to travel 4 cm:                       348
   pulses on each engine to a 90 degrees rotation:992

   The above physical quantities vary with the size of the wheels and the distance between the drive
   wheels.   */
```

```
#define triggercentrale 9      // trigger del sensore centrale
#define echocentrale 8         // echo del sensore centrale
#define triggerdestro 7        // trigger del sensore di destra
#define echodestro 6           // echo del sensore di destra
#include <Stepper.h>           // richiama la libreria per la gestione dei motori passo passo
Stepper stsinistro(32, 5, 3, 4, 2); // numero di impulsi per un giro dell'albero motore (32)
/*ed elenco delle porte di stsinsitro, in sequenza di attivazione
  Nota:  Le porte 5, 4, 3 e 2 di Arduino sono collegate, rispettivamente,
  ai pin In1, In2, In3 ed In4 del driver stsinistro, ma la sequenza di attivazione deve essere
  In1, In3, In2 ed In4  per cui la dichiarazione delle porte e' 5, 3, 4, e 2  */

Stepper stdestro (32, 14, 16, 15, 17); // numero di impulsi per un giro dell'albero motore (32)
// ed elenco delle porte di stdestro, in sequenza di attivazione
/*  nota: le porte 14, 15, 16 e 17 di Arduino (le porte analogiche da A0 ad A3) sono collegate,
  rispettivamente, ai pin In1, In2, In3 ed In4 del driver stdestro, ma la sequenza di attivazione
  deve essere In1, In3, In2 ed In4  per cui la dichiarazione delle porte e' 14, 16, 15, e 17  */
//
// variabili per la gestione dei due motori
int impulsidx            = 0;   // variabile con il numero di impulsi per il motore destro
int impulsisx            = 0;   // variabile con il numero di impulsi per il motore sinistro
int valoredifor          = 0;   // variabile utilizzata nella routine di azionamento motori
```

Arduino: fuga dal labirinto – maze escape

```
int impulsiperstep       = 348; // impulsi per percorrere 4 centimetri
int correttoredx         = 0;   // impulsi (segnati) per correggere la traiettoria del motore dx
int mxdx                 = 0;   // gestore del segno del motore dx nel ciclo di azionamento motori
int mxsx                 = 0;   // gestore del segno del motore sx nel ciclo di azionamento motori
int velocita             = 0;   // variabile per la gestione della velocita' dei motori
float cmcentrale         = 0;   // variabile di memorizzazione della distanza davanti al carrello
float cmdestra           = 0;   // variabile di memorizzazione della distanza a destra del carrello
int trigger              = 0;   // variabile utilizzata nella routine di lettura della distanza
int echo                 = 0;   // variabile utilizzata nella routine di lettura della distanza
float cm                 = 0;   // variabile utilizzata nella routine di lettura della distanza
int avanzamento          = 0;   // indice utilizzato nella routine di avanzamento
int impulsi              = 0;   // zona di lavoro, utilizzata nella routine di avanzamento
int correttore           = 45;  // variabile  di correzione del numero di impulsi sul motore dx,
// usata per compensare le inerzie meccaniche
int semaforodietrofront  = 0;   // se = 1 significa che ha appena fatto un dietrofront
//
//*************** routine di azionamento dei motori del carrello ***************
// ************************* cart driving routine ****************************
//
void avanti (void)
{
   valoredifor = max (abs(impulsidx), abs(impulsisx));  // il valore assoluto maggiore tra dx e sx
  mxdx = -1;               // imposta una rotazione oraria per il motore dx (per avanzare)
  mxsx = 1;                // imposta una rotazione antioraria per il motore sx(per avanzare)
  if (impulsidx < 0)       // se la ruota di dx deve indietreggiare
    mxdx = 1;              //imposta una rotazione antioraria per il motore di destra
  if (impulsisx < 0)       // se la ruota di sinistra deve indietreggiare
    mxsx = -1;              //imposta una  rotazione oraria per il motore di sinistra
  for (impulsi = 0; impulsi <= valoredifor; impulsi = impulsi + 1)
  {
    stdestro.step(mxdx);           // lancia un impulso al motore di destra
    if (impulsidx == 0)            // se si sono esauriti gli impulsi a destra
      mxdx = 0 ;                   // conclude il movimento del motore di destra
    stsinistro.step (mxsx);        // lancia un impulso al motore di sinistra
    if (impulsisx == 0)            // se si sono esauriti gli impulsi a sinistra
      mxsx = 0 ;                   // conclude il movimento del motore di sinistra
    impulsidx = impulsidx + mxdx;  // diminuisce il numero degli impulsi residui a destra (mxdx
// ha sempre segno contrario a impulsidx)
    impulsisx = impulsisx - mxsx;  //diminuisce il numero di impulsi residui a sinistra
    semaforodietrofront = 0;       // azzera il semafro dietrofront per segnalare che l'ultimo
// avanzamento non e' stato un dietrofornt
   }
}
//
// ************* routine di rilevamento delle distanze dalle pareti ********************
// ********************distance detecting routine ***************************************
//
void sensore (void)                 // routine di rilevamento della distanza dalla parete
{
  digitalWrite(trigger, LOW);       // disattiva il lancio del fascio di ultrasuoni
  delayMicroseconds (2);
  digitalWrite(trigger, HIGH);      // attiva il lancio del fascio di ultrasuoni
  delayMicroseconds(10);            // attende 10 microsecondi (il tempo del modulo HC-SR04)
  digitalWrite(trigger, LOW);       // disattiva il lancio del fascio di ultrasuoni
  cm = pulseIn(echo, HIGH) / 58.0;  // rileva il segnale di ritorno e lo converte in centimetri
}
//
// *********** routine per la rilevazione distanza avanti e destra *************
// *****************detect distance in front and on the right ******************
//
void rilevadistanze (void)
{
  trigger = triggercentrale;
  echo = echocentrale;
  sensore ();
  cmcentrale = cm;
  trigger = triggerdestro;
  echo = echodestro;
  sensore ();
  cmdestra = cm;
}
//
// ********** routine di orientamento del carrello di 90 gradi a destra  ************
//  ***************** turn cart 90 degrees on the right ****************************
//
void giraadestra ()
{
  if (!(semaforodietrofront == 1)) // se non ha appena fatto un dietrofront
```

```
  {
    impulsidx = 870;
    impulsisx = 870;
    avanti ();      // prosegue per circa 10 cm, in modo che il carrello superi il punto di svolta
  }
  impulsidx =  -990 - correttore; // il correttore compensa l'inerzia meccanica del motore destro
  impulsisx = 990;
  avanti ();        // fa girare il carrello di 90 gradi
  impulsisx = 0;
  impulsidx = correttore; // Il correttore compensa l'inerzia meccanica del motore di destra
  avanti ();              //compensa la inerzia meccanica del motore destro
  impulsidx = 1640;
  impulsisx = 1640;
  avanti ();  // prosegue per 19 cm, in modo da superare la curva
}
//
//****************** routine di inversione di marcia ************************
//**************************turn and go back routine ************************
//
void dietrofront ()
{
  impulsidx = -1980 - correttore;  // impulsi per una rotazione completa; il correttore compensa
// l'inerzia meccanica della ruota destra
  impulsisx = 1980;
  avanti ();
  impulsidx = correttore;  // controcompensa l'inerzia meccanica della ruota destra
  impulsisx = 0;
  avanti ();
  semaforodietrofront = 1;
}
//
// ************ routine di avanzamento di 4 cm, con controllo distanza dalle pareti **************
// ********************** 4 cm in advance routine, with wall distance control *******************
//
void avanzaecontrolla (void)
{
  rilevadistanze ();
  correttoredx = 0;          // azzera preventivamente il correttore di impulsi sulla ruota destra
  if (cmdestra < 5)          // se il carrello e' troppo a destra
    correttoredx = 200;
  if (cmdestra > 8)          // se il carrello e' troppo a sinistra
    correttoredx = -200;
  if (!(correttoredx == 0))  // se deve essere riallineato il carrello
  {
    impulsidx = correttoredx;
    impulsisx = correttoredx * -1;
    if (impulsidx < 0)
      impulsidx = impulsidx - correttore;   // corregge gli impulsi su motore dx per compensare
// le inerzie meccaniche
    avanti ();                // orienta il carrello per recuperare la linea mediana del corridoio
    impulsidx = 500;          // impulsi al motore dx, per un avanzamento di 6 cm in diagonale
//(4 cm sul cateto piu' lungo)
    impulsisx = 500;          // impulsi al motore sinistro, per un avanzamento di 6 cm
    avanti ();                // fa avanzare il carrello di 4 centimetri sul cateto piu' lungo
    if (correttoredx < 0)
      correttoredx = correttoredx + 23;  // modifica il correttoredx per correggere le distonie
// che hanno portato il carrello a non essere parallelo al muro
    else
      correttoredx = correttoredx - 23;
    impulsisx = correttoredx; // inverte il senso, in modo da riorientare la direzione del carrello
    impulsidx = correttoredx * -1;
    if (correttoredx < 0)
      impulsidx = impulsidx + correttore;   // correge gli impulsi su motore dx per compensare
// le inerzie meccaniche
    avanti ();                // riorienta il carrello per mantenere la linea mediana del corridoio
  }
  else                        // se la direzione non deve essere modificata
  {
    impulsidx = 300;          // impulsi al motore destro, per un avanzamento di 3,5 cm
    impulsisx = 300;          // impulsi al motore sinistro, per un avanzamento di 3,5 cm
    avanti ();
  }
}
//
//
void setup()
{
  delay (3000);          // attende 3 secondi prima di avviare il programma
```

```
  Serial.begin(9600);    // inizializza la porta di comunicazione con il monitor seriale
  pinMode(triggercentrale, OUTPUT);    // la porta 9 (il trigger centrale) e' in output
  pinMode(echocentrale, INPUT);        // la porta 8 (l'echo centrale) e' in input
  pinMode(triggerdestro, OUTPUT);      // la porta 7 (il trigger destro) e' in output
  pinMode(echodestro, INPUT);          // la porta 6 (l'echo destro) e' in input
  velocita = 900;      // imposta un'alta velocita' di rotazione dei motori (900 impulsi al secondo)
  stdestro.setSpeed(velocita);   // imposta la velocita' del motore destro
  stsinistro.setSpeed(velocita); // imposta la velocita' del motore sinistro
  //
  //  ************  test motori passo passo ******************
  // ************ un giro in avanti motore sx e motore dx
  // impulsidx = 2048;
  // impulsisx = 2048;
  //  avanti ();
  //
  // **********************test routine avanti
  // impulsidx = 500;
  // impulsisx = -500;
  // avanti ();
  //
  // ******************* test routine di misurazione della distanza
  // rilevadistanze ();
  // Serial.print ("centrale:");
  // Serial.print (cmcentrale);
  // Serial.print ("  destra:");
  // Serial.println (cmdestra);
  //
  // ********** test routine gira a destra
  // giraadestra ();
  //
  // ************* prova routine dietrofront
  // dietrofront();
  //
  // ******************* prova routine avanzaecontrolla
  // for (int i = 0; i < 100; i++)
  //  avanzaecontrolla ();
}
//
//
void loop()
{
  rilevadistanze ();
  if (cmdestra > 30)              // se c'e' un varco a destra  - if there is a way on the right
  {
    giraadestra ();               // orienta a destra il carrello - Turn cart on the right
    return;
  }
  if (cmcentrale < 8)             // se si trova in un vicolo cieco (o deve girare a sinistra)
//                                   if is on a blind street (or must go on the left)
  {
    dietrofront ();               // compie una rotazione di 180 gradi - turn and go back
    return;
  }
  avanzaecontrolla ();            // avanza di 4 cm controllando al distanza dalla parete destra
//                                   Advances 4 cm and check distance from the right wall
}
```