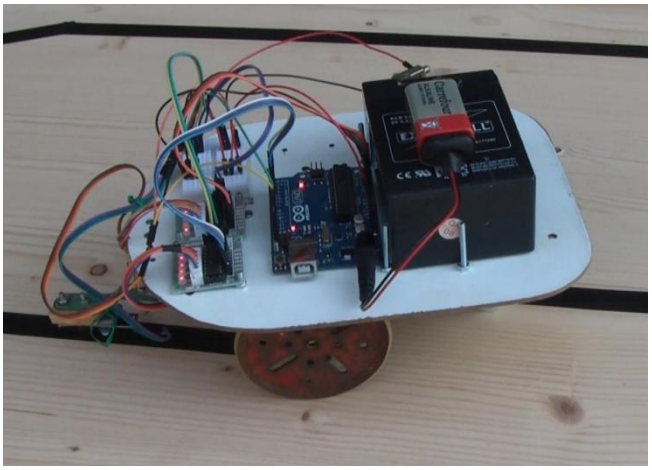
F- Snoopy: carrello segugio - the hound cart (some notes at section end)



Snoopy e' un carrello mosso da due motori a 4 poli. Un sensore di traccia posizionato su di un braccio mobile fissato alla parte anteriore rende snoopy un segugio imbattibile. Trova la traccia (una striscia nera dipinta o comunque posizionata su di un fondo chiaro) e la segue.

Il sensore e' pilotato da un servomotore Sg90 che fa ruotare di 180 gradi il braccio su cui e' fissato e gli consente di ritrovare la traccia nel momento in cui dovesse perderla. Il carrello e' mosso da due motori a quattro poli pilotati dai relativi driver.

La parte piu' interessante del progetto sembra essere quella relativa al calcolo del percorso dopo che il sensore ha individuato la traccia.

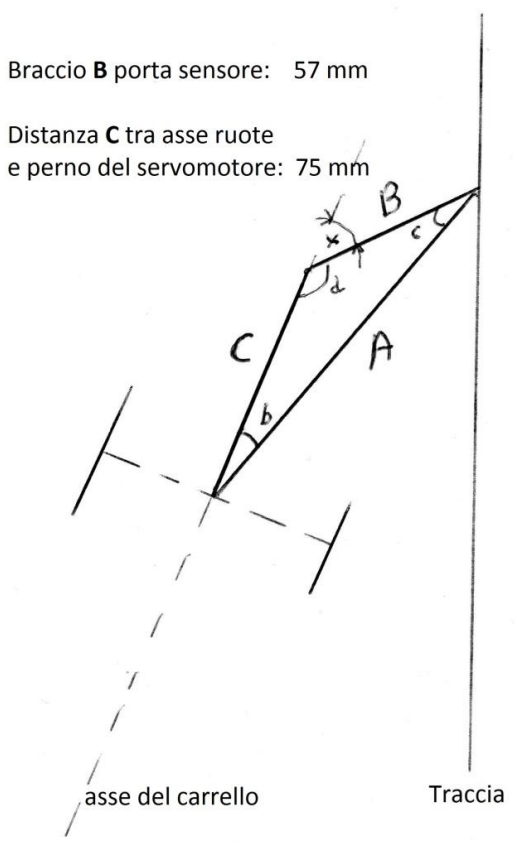
Il carrello infatti non conosce l'orientamento della traccia, ma sa solo la sua posizione (a destra o a sinistra del suo asse longitudinale) e l'angolazione (in gradi sessagesimali) del braccio porta sensore rispetto al suo asse longitudinale. Queste due informazioni, unite alla lunghezza del braccio porta sensore (nota) ed alla distanza tra l'asse delle ruote ed il perno del servomotore (anch'essa nota) consente di calcolare le informazioni necessarie al movimento del carrello ed al suo mantenimento sulla traccia.

In realta' il funzionamento del prototipo si e' rivelato non particolarmente entusiasmante poiche' quando la traccia non segue una linea retta il carrello si ferma per cercarla ed i tempi di stop, dipendenti da quanto la traccia e' distante e dalla velocita' di movimento del braccio, sono sempre comunque dell'ordine di qualche secondo. E' forse possibile un miglioramento utilizzando almeno tre sensori ed eliminando quindi la necessita' del servomotore che, per quanto efficiente, non e' mai particolarmente veloce.

Qualche grandezza fisica e qualche calcolo (vedi anche il disegno e lo schema di calcolo nella prossima pagina)

diametro ruote motrici: 75 mm
circonferenza ruote motrici 235 mm
distanza tra sensore e asse servomotore: 57 mm
distanza tra asse servomotore e asse ruote: 75 mm
impulsi per rotazione competa della ruota: 2048
avanzamento ogni 100 impulsi: 11,47 mm
rotazione per 100 impulsi: 7 gradi e 20 centesimi di grado
impulsi su di una ruota per rotazione di 1 grado dell'asse del carrello: 14

Quando il sensore trova la traccia si dispone del valore ***x*** (tra l'asse longitudinale del carrello ed il braccio **B** del sensore). Questa informazione, unita alla lunghezza del braccio portasensore **B** (57 mm) ed alla distanza tra l'asse del servo e l'asse delle ruote motrici **C** (75 mm) ci consente di calcolare attraverso le normali formule trigonometriche, la distanza **A** tra l'asse del carrello e la traccia nera e l'angolo di rotazione ***b*** che deve essere impresso al carrello per consentirgli di raggiungere la linea nera percorrendo la distanza **A.**



Valore dell'angolo ***a*** in radianti:

a = (180 – ***x***) * 6,28/360 = (180-**x**) * 0,01744

la funzione cos() di wiring consente di calcolare il coseno partendo dai radianti per cui:

coseno di *a* = cos ((180-**x**) * 0,01744)

applicando la formula del coseno:

$\mathbf{A}^2 = \mathbf{C}^2 + \mathbf{B}^2 - 2*\mathbf{B}*\mathbf{C}*\cos a$ si calcola **A** (in mm)

$\mathbf{A} = \text{sqrt}\ (57^2+75^2-2*57*75*\cos((180-\mathbf{x})*0{,}01744))$

e quindi:

A = sqrt (8874 – 8550*cos((180-**x**)*0,01744))

Per calcolare l'angolo ***b*** in gradi sessagesimali, non essendo stata trovata un'adeguata funzione trigonometrica in wiring (forse esiste, ma non e' stata trovata), si calcola il coseno e lo si utilizza per scorrere una tabella di conversione "coseno -> angolo" limitata a soli 90 valori (i 90 gradi senza minuti e secondi) e memorizzata nel programma.

Il coseno di ***b*** e' ottenuto dalla formula: $(\mathbf{C}^2+\mathbf{A}^2-\mathbf{B}^2) / (2*\mathbf{A}*\mathbf{C})$ e quindi:
coseno di ***b*** = $(75^2+\mathbf{A}^2-57^2) / (2*\mathbf{A}*75) = (\mathbf{A}^2+2376) / (\mathbf{A}*150)$

Una volta ottenuto il coseno e' poi possibile, tramite la tabella dei coseni memorizzata nel programma, risalire al valore dell'angolo in gradi sessagesimali e quindi calcolare gli impulsi da imprimere alle ruote per far ruotare il carrello verso la migliore angolazione di impatto.

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

Snoopy is a cart driven by two stepper motors. A track sensor, positioned on a movable arm fixed at cart front makes it a great hound. Find the track (a black stripe positioned on a light background) and follows it.

The track sensor is positioned on an arm driven by a servo motor SG90 which rotates 180 degrees. Cart is driven by two stepper motors.

The project most interesting part seems to be the path calculation, when sensor detects a trace.

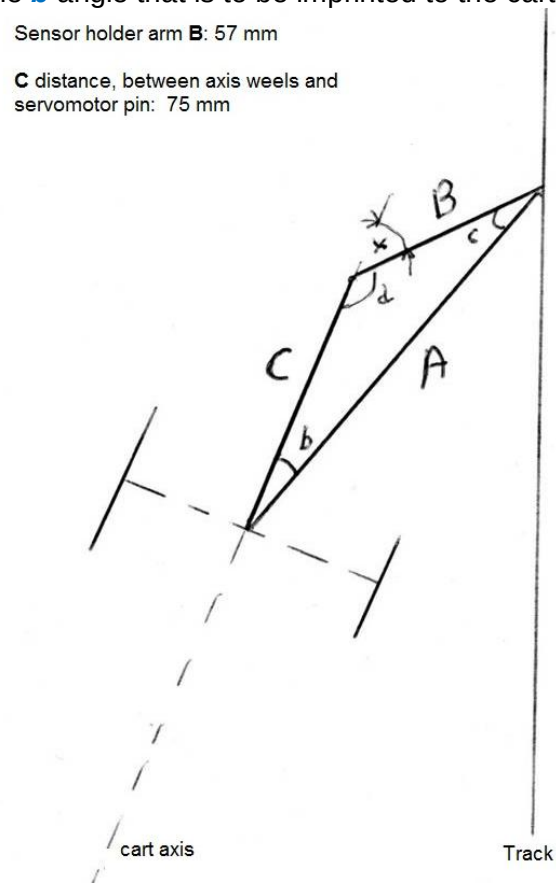
Arduino don't know track orientation but only the arm position (on the right or on the left) and angle respect its longitudinal axis. These two information, combined with the sensor arm length and the distance between the wheel axis and the servomotor pin is used to calculate information about cart movement and its run on track.

The prototype use was revealed not particularly exciting because, when track does not follow a straight line, the cart stops to find track and the stop times depends on track distance and arm speed. Is maybe possible an improvement by using three track sensors and eliminating the servo motor, never particularly performant.

Some physical data and some calculations (see also design and calculation on the next picture)

- driving wheels diameter : 75 mm
- driving wheels circumference: 235 mm
- distance between sensor and servo motor axis: 57 mm
- distance between servo motor axis and wheel axis: 75 mm
- pulses for a complete rotation wheel: 2048
- street traveled for 100 pulses: 11,47 mm
- rotation for 100 pulses: 7 degrees and 20 hundredths
- pulses for 1 degree wheel rotation: 14

Looking the next picture, when sensor finds the track, we have the ***x*** angle value (between the cart longitudinal axis and the arm **B**). This information, combined with the sensor carrier arm B lenght (57 mm) and distance **C** between the servo axis and wheels axis (75 mm) enables us to calculate, through the normal trigonometric formulas, the distance **A** between cart axis and black track, and the ***b*** angle that is to be imprinted to the cart to reach the black line along the distance **A**.



angle "***a***" value, in radians:

a = (180 – ***x***) * 6,28/360 = (180-**x**) * 0,01744

the wiring ***cos()*** function can calulate a cosine from radians, so:

a cosine = cos ((180-**x**) * 0,01744)

applying the cosine formula:

$\mathbf{A^2} = \mathbf{C^2} + \mathbf{B^2} - 2*\mathbf{B}*\mathbf{C}*\cos a$ can calculate **A** (in mm):

A = sqrt (57^2+75^2–2*57*75*cos((180-**x**)*0,01744))

And so:

A = sqrt (8874 – 8550*cos((180-**x**)*0,01744))

To calculate the ***b*** angle in degrees, since was not found any trigonometric function in wiring language (perhaps exists, but it has not been found), we calculates first the ***b*** cosine, and we use it to find the ***b*** angle, through a conversion matrix "cosine - > angle", stored in program.

the ***b*** cosine is obtained by this formula: ($\mathbf{C^2}+\mathbf{A^2}-\mathbf{B^2}$) / (2***A*C**) an so:

b cosine = ($75^2+\mathbf{A^2}-57^2$) / (2***A***75) = ($\mathbf{A^2}$+2376) / (**A***150)

Once obtained the cosine, you can, by the cosines stored matrix, find the angle value in degrees, and then calculate pulses to be imparted to wheels, to rotate cart on the best impact angle.

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)

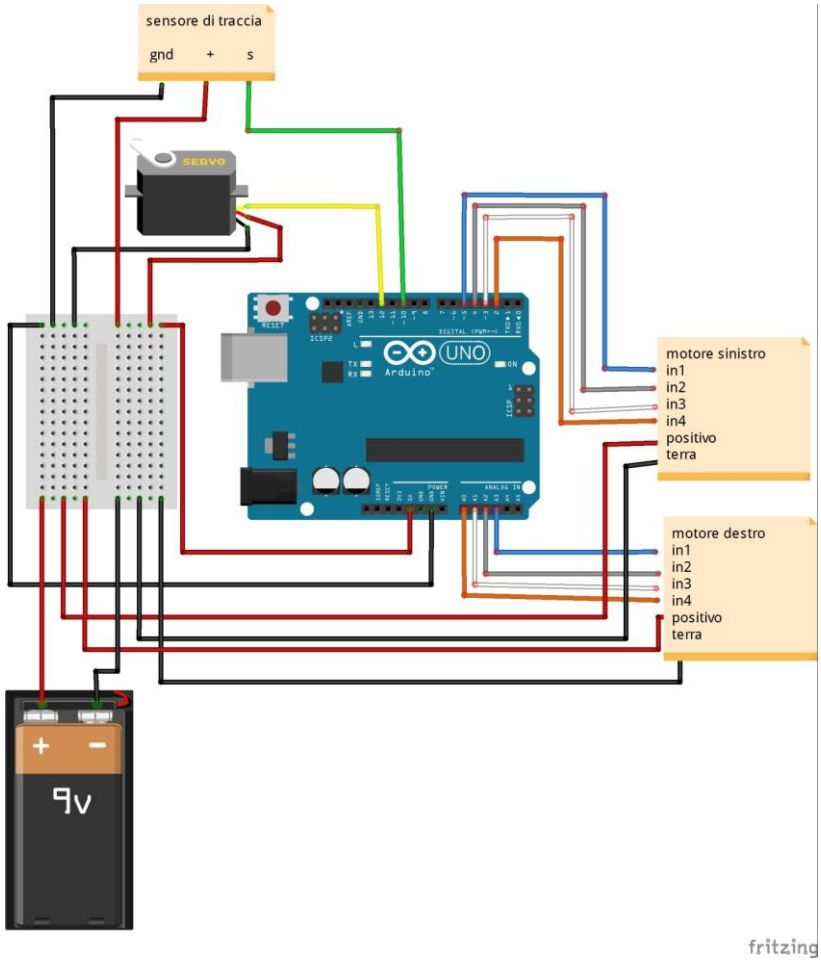
For any questions or suggestions about this note (and on its english translation), please write to giocarduino@libero.it (simple words and short sentences, please)

Materiali

Oltre ad alcuni pezzi di meccano e alcune parti in legno, da sagomare opportunamente, sono necessari:

- due motori passo passo con i relativi driver
- un servomotore tipo sg90
- un sensore di traccia (vedi anche esercizio 30)
- una batteria da 9 volt
- una batteria da almeno 6 volt, di grande capacita' ed in grado di azionare i motori
- una breadborad di piccole dimensioni

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.
*
* Snoopy e' un carrello mosso da due motori a 4 poli. Un sensore di traccia posizionato su di un
* braccio mobile fissato alla parte anteriore consente al carrello di trovare e seguire una traccia
* (una striscia scura posizionata su di un fondo chiaro. Il sensore e' pilotato da un servomotore
* Sg90 che fa ruotare di 180 gradi il braccio su cui e' fissato e gli consente di ritrovare la
* traccia nel momento in cui dovesse perderla.
*
* Dettagli sulle dimensioni del braccio e del carrello, oltre che una approfondita disamina dei
* calcoli necessari a mantenre il carello sulla traccia, sono presenti nella nota di questo
* progetto, repribile in:
* http://giocarduino.altervista.org/arduino-prototipi/snoopy.htm )
*
*-----------------------------------------------------------------------------------------------
* Warning: cut&paste from PDF to IDE loses formatting. to restore it press CTRL + T.
*
* Snoopy is a cart driven by two stepper motors. A track sensor, positioned on a movable arm fixed
* to front, allows the cart to find and follow a track (a dark stripe on a clear background). The
* sensor is driven by a SG90 servomotor, which can rotate 180 degrees the arm on which is fixed,
* and allows it to regain track when lose it.
*
* You can found datai abouti this project in:
* http://giocarduino.altervista.org/arduino-prototipi/snoopy.htm
*-----------------------------------------------------------------------------------------------
*/
#define pinsensore 10     // assegna la porta 10 al sensore di traccia
#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 stsinistro, in sequenza di attivazione. 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  - pulses for one revolution of the crankshaft (32) and list of pins of stsinistro (the left
stepper motor), in  activation sequence. The arduino pins 5, 4, 3 and 2 are connected respectively
to stsinistro driver pins In1, In2, In3 and In4, but the activation sequence must be In1, In3, In2
and In4, for which the declaration of pins is 5, 3, 4, and 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  - pulses for one revolution of the crankshaft (32)
and list of pins of stdestro (the right stepper motor), in  activation sequence. The arduino pins
14,15,16 and 17 (the analog pins form A0 to A3) are connected respectively
to stdestro driver pins In1, In2, In3 and In4, but the activation sequence must be In1, In3, In2
and In4, for which the declaration of pins is 14, 16, 15, e 17.
*/
#include <Servo.h>    // richiama la libreria di gestione del servomotore
Servo pippo;          // crea il servo oggetto "pippo" da utilizzare per riferire il servomotore
//
// variabili per la gestione dei due motori - variables to stepper motors management
int impulsidx  = 0;   // variabile in cui viene inserito il numero di impulsi per il motore destro
int impulsisx  = 0;   // variabile in cui viene inserito il numero di impulsi per il motore sinistro
int valoredifor= 0;   // variabile utilizzata nel ciclo di for della routine di azionamento motori
int impulsi    = 0;   // gestore del ciclo di for nella routine di azionamento motori motori
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
//
// variabili per la gestione del servomotore - variabiles to servo management
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  = 0;  // step di incremento o decremento dei gradi del servomotore
//
// variabili per la gestione del sensore di traccia - variable to track sensor management
int statotraccia= 0;  // variabile di memorizzazione dello stato del sensore di traccia
//
// variabili per il calcolo del percorso - variables to path calculation
int gradirotazione = 0;// gradi di rotazione dell'asse del carrello
double lunghezza = 0;  // percorso per mettere il carrello a cavallo della traccia (decimali)
```

```
int percorso     = 0;  // percorso per mettere il carrello a cavallo della traccia (intero)
int angolox      = 0;  // angolazione in gradi sessagesimali del braccio porta sensore
//                        rispetto all'asse del carrello
double cosenodia = 0;  // variabile per la memorizzazione del coseno di a (angolo tra asse
//                        carrello e braccio porta sensore)
double calcolointermedio = 0; // variabile utilizzata per calcoli intermedi
double cosenodib = 0; // variabile per la memorizzazione del coseno di b (angolo opposto
                          // al braccio portasensore)
int angolob      = 0;  // indice di scorrimento tabella coseno, coincidente con l'angolo b
int cosenoint    = 0;  // cosenodib, moltiplicato per 10000 e privato dei decimali
int semaforoasse = 0;  // semaforo che indica la situazione dell'asse delle ruote rispetto alla
//           traccia:  0 = l'asse e' fuori dalla traccia; 1 = l'asse e' a cavallo della traccia
int coseno [91] = { // tabella di conversione da coseno*10000 ad angolo sessagesimale (l'angolo
//  e' l'indice di scorrimento) il coseno 10000 corrisponde a 0 gradi, il coseno 7071 corrisponde a
// 45 gradi ed il coseno 0 corrisponde a 90 gradi - cosine-> angle matrix conversion: data are
// cosine* 1000 and the relative index is the angle
  10000,
  9998,
  9994,
  9986,
  9976,
  9962,
  9945,
  9925,
  9903,
  9877,
  9848,
  9816,
  9781,
  9744,
  9703,
  9659,
  9613,
  9563,
  9511,
  9455,
  9397,
  9336,
  9272,
  9205,
  9135,
  9063,
  8988,
  8910,
  8829,
  8746,
  8660,
  8572,
  8480,
  8387,
  8290,
  8192,
  8090,
  7986,
  7880,
  7771,
  7660,
  7547,
  7431,
  7314,
  7193,
  7071,
  6947,
  6820,
  6691,
  6561,
  6428,
  6293,
  6157,
  6018,
  5878,
  5736,
  5592,
  5446,
  5299,
  5150,
  5000,
  4848,
```

```
  4695,
  4540,
  4384,
  4226,
  4067,
  3907,
  3746,
  3584,
  3420,
  3256,
  3090,
  2924,
  2756,
  2588,
  2419,
  2250,
  2079,
  1908,
  1736,
  1564,
  1392,
  1219,
  1045,
  872,
  698,
  523,
  349,
  175,
  0
};
//
//*************** routine di azionamento dei motori del carrello ***************
//****************** stepper motors management routine **********************
//
void avanti (void)          {
stdestro.setSpeed(velocita);    // imposta la velocita' del motore destro
stsinistro.setSpeed(velocita); // imposta la velocita' del motore sinistro
  valoredifor = max (abs(impulsidx), abs(impulsisx));//rileva il maggior valore assoluto tra dx e sx
  // Serial.print (" valoredifor ="");  // Traccia di debug
  // Serial.println (valoredifor);
  mxdx = 1;                // preimposta un avanzamento positivo per il motore di destra
  //                          (senso orario, per fare avanzare il carrello)
  mxsx = -1;               // preimposta un avanzamento negativo per il motore di sinistra (senso
 //                          antiorario, per fare avanzare il carrello)
  if (impulsidx < 0)       // se la ruota di dx deve indietreggare
    mxdx = -1;             // imposta un avanzamento negativo per il motore di destra
  if (impulsisx < 0)       // se la ruota di sinistra deve indietreggiare
    mxsx = 1;              // imposta un avanzamento positivo per il motore di sinistra (per farla
  //                          indietreggiare)
  for (impulsi = 0; impulsi <= valoredifor; impulsi = impulsi + 1)
  {
    stdestro.step(mxdx);

    if (impulsidx == 0)    // se si sono esauriti gli impulsi a destra
    {
      mxdx = 0 ;           // conclude il movimento del motore di destra
    }
    stsinistro.step (mxsx);

    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
    impulsisx = impulsisx + mxsx;  //diminuisce il numero di impulsi residui a sinistra
    //                               (mxsx ha sempre segno contrario a impulsisx)
    //  Serial.print (" impulsidx ="");  // traccia di debug
    //  Serial.print (impulsidx);
    //  Serial.print (" impulsisx ="");
    //  Serial.print (impulsisx);
    //  Serial.print (" impulsi ="");
    //  Serial.println (impulsi);
  }
}
//
// *************routine di rilevamento della traccia***************
// ***************** track finding routine *******************
//
```

Arduino: Snoopy il carrello segugio – the hound cart

```
void sensore (void)
{
  //  Serial.println("routine sensore");  // traccia di debug
  gradi = posprec;
  incremento = 1;
  if (pos <= posprec)
    incremento = -1;
  //  Serial.print("gradi=");  // traccia di debug
  //  Serial.print(gradi);
  //  Serial.print(" incremento=");
  //  Serial.print(incremento);
  //  Serial.print(" pos=");
  //  Serial.println(pos);
  for (posprec = gradi; (posprec < pos || posprec > pos); posprec = posprec + incremento) // sposta
 //    di 1 grado per volta, da posprec a pos, l'angolazione del braccio
  {
    //   Serial.print("pos=");  // traccia di debug
    //   Serial.println(pos);
    //   Serial.print(" posprec=");
    //   Serial.println(posprec);
    pippo.write(posprec); // indirizza il perno alla posizione desiderata, memorizzata in 'posprec'
    delay(20);       // attende 20ms per consentire al servomotore di raggiungere la posizione
    statotraccia = digitalRead (pinsensore);  //rileva il segnale del sensore
    //    Serial.print("statotraccia=");
    //    Serial.println(statotraccia);       // visualizza lo stato del sensore
    if (statotraccia == HIGH)       // se ha trovato la banda nera
      pos = posprec + incremento;   //interrompe la ricerca
  }
}

void calcolapercorso (void)         // routine di calcolo del percorso (angolazione e distanza)
{
//
//*****************  calcolo della lunghezza del percorso *************************
// ******************** path lenght calculation routine ***************************
//
  cosenodia = cos((180 - angolox) * 0.01744);
  // Serial.print (" cosenodia =");
  // Serial.println (cosenodia);
  calcolointermedio = (8550 * cosenodia);
  calcolointermedio = 8874 - calcolointermedio;
  //  Serial.print (" calcolointermedio =");
  //  Serial.println (calcolointermedio);
  lunghezza = sqrt (calcolointermedio);
  percorso = lunghezza;   // toglie i decimali
  //  Serial.print (" percorso =");
  //  Serial.println (percorso);
//
//****************** calcolo dell'angolazione del carrello ************************
//********************** cart angle calculation routine ***************************
//
  cosenodib = ((percorso * percorso) + 2376) / (percorso * 0.0150); // coseno moltiplicato per 10000
  //  Serial.print (" cosenodib =");
  //  Serial.println (cosenodib);
  for (angolob = 1; angolob <= 90; angolob++)  //scorre la tabella cosenodib per trovare il
//      coseno dell'angolo b
  {
    if ((cosenodib >= coseno [angolob]) && ( cosenodib <= coseno [angolob - 1]))
    {
      //    Serial.print (" angolob =");
      //    Serial.println (angolob);
      break;
    }
  }
}
//
void setup()
{
  delay (3000);         // attende 3 secondi prima di avviare il programma
  pippo.attach(12);     // assegna il servomotore portasensore alla porta 12
  pippo.write (90);     // posiziona il sensore in linea con il senso di marcia
  delay (200);          // attende che il servomotore sia in posizione
  posprec = 90;         // memorizza la posizione del sensore
  pinMode(pinsensore, INPUT); // definisce la porta digitale 10 (il sensore di traccia) come
  //        porta di input
  // Serial.begin(9600);  // inizializza la comunicazione con il monitor seriale (per debug)
  velocita = 700;         // imposta un'alta velocita' di rotazione dei motori passo passo
//                           (700 impulsi al secondo)
```

```
//
  // *********** test routine calcolapercorso  *****************
  //  angolox = 60;
  //  calcolapercorso ();
  //  Serial.println (percorso);
  //  Serial.println (angolob);

  //  ************  test motori passo passo *****************
  // ************ mezzo giro in avanti motore sx e motore dx
  // impulsidx = 1024;
  // impulsisx = 1024;
  //  avanti ();
}
//
void loop()
{
  statotraccia = 0;
  pos = 90;                  // posiziona il sensore in linea con il senso di marcia
  //  Serial.println("prima traccia loop");
  sensore ();                //verifica se il carrello e' a cavallo della traccia
  if (statotraccia == HIGH) // se il sensore vede la banda nera ed e' in asse con essa
  {
    impulsisx = 300;
    impulsidx = 300;
    avanti();                // avanza di circa 42 mm
  }
  if (statotraccia == LOW)  // se il carrello non e' in asse con la traccia
  {
    //  Serial.println("seconda traccia loop");
    pos = 15;                // Verifica se la traccia e' a sinistra
    sensore();
    if (statotraccia == HIGH)// se la traccia e' a sinistra
    {
      angolox = 90 - posprec;// determina l'angolo tra l'asse del carrello ed i braccio portasensore
      calcolapercorso ();    // calcola il percorso (angolazione e distanza) per portare l'asse delle
   //                           ruote a cavallo della traccia
      impulsidx = angolob *  7; // impulsi sulla ruota dx necessari ad imprimere al carrello
   //                           l'angolazione di incrocio con la traccia
      impulsisx = angolob * -7; // impulsi negativi a sinistra per far muovere i carrello a sinistra
      avanti ();                // ruota il carrello in direzione della traccia
      impulsidx = percorso / 0.115; // calcolo impulsi necessari a raggiungere la traccia
      if (impulsidx > 300)      // se gli impulsi prevedono un tragitto superiore ai 4 cm,
        impulsidx = 300;        // forza un max di 4 cm (300 impulsi) per poi ripetere la verifica
      //                        dei sensori
      impulsisx = impulsidx;
      if (semaforoasse == 0)    // Se l'asse delle ruote non e' gia' a cavallo della traccia
      {
        avanti ();              // porta l'asse delle ruote a cavallo della traccia
        semaforoasse = 1;       // segnala che al prossimo giro bastera' girare il carrello per
        //                      metterlo in linea con la traccia, senza farlo avanzare
        //                      poiche' l'asse delle ruote e' gia' sulla traccia
      }
      else
        semaforoasse = 0;       // segnala che al prossimo giro bisognera' anche portare l'asse
      //                        delle ruote sulla traccia
    }
  }
  if (statotraccia == LOW)     // se non e' stata trovata traccia a sinistra
  {
    pos = 165;                 // verifica se la traccia e' a destra
    sensore ();
    if (statotraccia == HIGH)  // se la traccia e' a destra
    {
      angolox = posprec - 90;  // determina l'angolo tra l'asse del carrello ed i braccio
      calcolapercorso ();      // calcola il percorso (angolazione e distanza) per portare l'asse a
                               //  cavallo della traccia
      impulsidx = angolob * -7; // impulsi sulla ruota dx necessari ad imprimere al carrello
                               //  l'angolazione di incrocio con la traccia
      impulsisx = angolob * 7;  // impulsi sulla ruota di sinistra per far girare il carrello a dx
      avanti ();
      impulsidx = percorso / 0.115; // calcolo impulsi necessari a raggiungere la traccia
       if (impulsidx > 300)     // se gli impulsi prevedono un tragitto superiore ai 4 cm,
        impulsidx = 300;        // forza un max di 4 cm (300 impulsi) per poi ripetere la verifica
       //                       dei sensori
      impulsisx = impulsidx;
      if (semaforoasse == 0)    // Se l'asse delle ruote non e'gia' a cavallo della traccia
      {
        avanti ();              // porta l'asse delle ruote a cavallo della traccia
```

```
        semaforoasse = 1;       // segnala che al prossimo giro bastera' girare il carrello per
        //                         metterlo in linea con la traccia, senza farlo avanzare
        //                         poiche' l'asse delle ruote e' gia' sulla traccia
      }
      else
        semaforoasse = 0;       // segnala che al prossimo giro bisognera' anche portare l'asse
   //                             delle ruote sulla traccia
     }
  }
  if (statotraccia == LOW)       // se nemmeno a destra e' stata trovatala traccia
  {
    impulsidx =  300;    impulsisx =  300;
    avanti ();// fa avanzare il carrello di 300 impulsi (circa 4 centimetri) alla ricerca della
traccia
  }
}
```