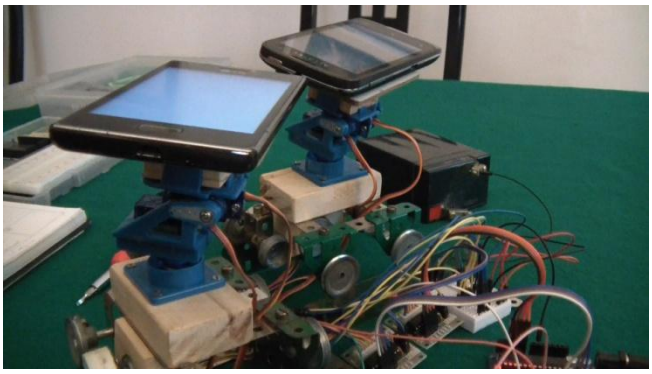
L – cinerama (some notes at end of this section)

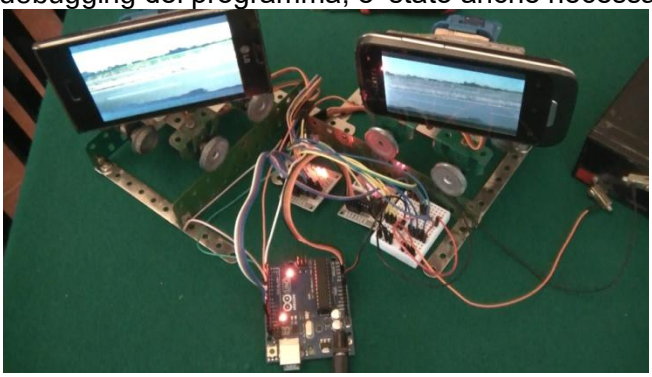


L'idea nasce dalla visita al padiglione della Corea, all'Expo. Padiglione nel quale erano presenti due bracci meccanici che reggevano due grandi schermi che, dopo svariate evoluzioni, proiettavano due film le cui immagini erano in sincronia temporale (un oggetto correva lungo il primo schermo, arriva al bordo, scompariva e subito dopo riappariva sul secondo schermo e continuava a correre).

Uno spettacolo affascinante, che di fatto ha attirato l'attenzione e suscitato meraviglia in centinaia di migliaia di persone.

In questo casalingo prototipo i due bracci ed il filmato non offrono, ovviamente, la medesima spettacolarita' dell'allestimento coreano, ma offrono, in scala minore, le medesime funzioni. I coreani hanno mostrato un impianto poderoso ma, al di la delle apparenze, di tecnologia non troppo innovativa, visto che con Arduino, quattro servomotori, due motori stepper e due telefonini obsoleti, si riesce a proporre qualcosa di simile (almeno nei principi).

In questo prototipo, oltre alla messa a punto delle parti meccaniche ed oltre alla scrittura ed al debugging del programma, e' stato anche necessario produrre un filmato adeguato alla bisogna.



In realta', come e' abbastanza facile comprendere, i filmati sono due, uno per lo schermo destro ed uno per lo schermo sinistro che, avviati contemporaneamente, proiettano le medesime immagini con alcuni studiati ritardi, per far si che le immagini delle auto in movimento o del panorama che scorre offrano l'idea di due differenti finestre sotto le quali passano, con i giusti intervalli di tempo, sia le auto che le immagini del panorama.

Una nota a parte merita l'impianto di alimentazione di tutto il complesso. Arduino non riesce a fornire l'energia necessaria per muovere contemporaneamente i quattro servomotori ed i due motori passo passo per cui e' stato necessario ricorrere ad un'alimentazione esterna, in questo caso una batteria da 6 volt e 4.2 ampere. L'alimentazione esterna e la contemporanea gestione degli impulsi a motori e servomotori e' stata resa possibile dall'unione dei poli negativi, come d'altronde appare chiaro dallo schema.

Nota: Questo esercizio 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 

The idea comes from a visit to the Korea Expo pavilion. An hall where there were two mechanical arms with two large screens that after several movements, showed a film whose images were synchronized (an object ran along the first screen, came to the edge and disappeared, reappearing immediately on the second screen).

A fascinating spectacle, which in fact has attracted the attention and aroused wonder on hundreds of thousands people.

This homemade prototype, arms and movie does not offer the same fascinating show, but offers, on a smaller scale, the same functions.

In this prototype, in addition to the development of the mechanical parts, was also necessary to produce a movie suitable at needs.

In reality, and as is quite easy to understand, the movies are two, one for the right screen and one for left-hand screen. These movies shows the same images with some delays, to offers the idea of two different windows under which pass, with the right intervals of time, both the cars that the images of the panorama.

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.

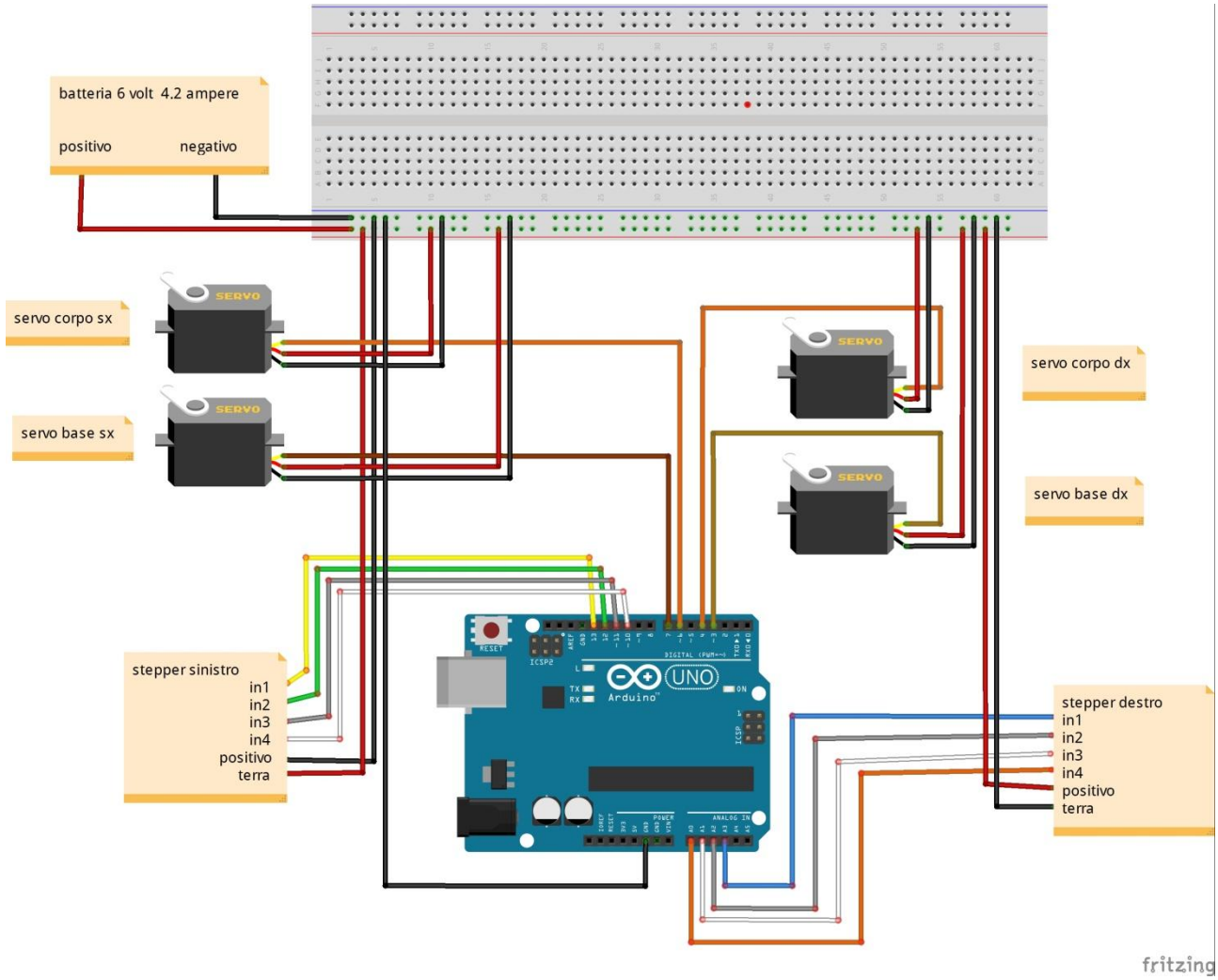
- Projects collection
- Movies on youtube
- About Arduino and components (italian; pdf will be downloaded in your download area
- Quick programming guide (almost english; pdf will be downloaded in your download area)

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 motori passo passo, con relativi driver
- Quattro servomotori sg90 montati su due snodi in plastica
- Alcuni (parecchi) pezzi di un vecchio meccano
- Due telefonini android, ormai obsoleti
- Due filmati adeguatamente montati
- Una batteria da 6 volt e 4.2 ampere

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    Programma per la gestione degli stepper e dei servo
* del cinerama. Tutte le mosse degli stepper e dei servomotori sono registrate nella tabella
* mosse; il programma si limita ad eseguire i movimenti indicati in ogni riga di detta tabella
*
*-----------------------------------------------------------------------------------------------
* Warning: cut&paste from PDF to IDE loses formatting. to restore it press CTRL + T.
* This program manages stepper and servo. All moves of stepper and servo are recorded in an array;
* the program only perform movements indicated in each row of said  array
*-----------------------------------------------------------------------------------------------
*
*/
#include <Servo.h>  // richiama la libreria di gestione dei servomotori
Servo basesx;       // crea il servo oggetto "basesx" da utilizzare nel programma.
Servo corposx;      // crea il servo oggetto "corposx" da utilizzare nel programma.
Servo basedx;       // crea il servo oggetto "basedx" da utilizzare nel programma.
Servo corpodx;      // crea il servo oggetto "corpodx" da utilizzare nel programma.
// in un programma possono essere creati, al massimo, otto servo oggetti

#include <Stepper.h>  // richiama la libreria con le routine di gestione del motore passo passo
Stepper stsinistro(32, 13, 11, 12, 10); // numero di impulsi per un giro dell'albero motore (32)
//  ed elenco delle porte di stdestro, in sequenza di attivazione
/*  nota:  Le porte 13, 12, 11 e 10 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' 13, 11, 12, e 10
*/
Stepper stdestro (32, 14, 16, 15, 17); // numero di impulsi per un giro dell'albero motore (32)
// ed elenco delle porte di stsinistro, 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
*/
//****************definzione delle variabili ***************************
int i                    = 0;   // indice di scorrimento della tabella mosse
int k                    = 0;   // indice del ciclo di for per il movimento dei motori e dei servo
int precbasesx           = 0;   // posizione precedente della base sinistra
int preccorposx          = 0;   // posizione precedente del corpo sinistro
int precbasedx           = 0;   // posizione precedente della base destra
int preccorpodx          = 0;   // posizione precedente del corpo sinistro
int movstsinistro        = 0;   // variabile di memorizzazione del movimento stepper sinistro
int movstdestro          = 0;   // variabile di memorizzazione del movimento stepper destro
int movcorposx           = 0;   // variabile di memorizzazione del movimento servo corpo sx
int movbasesx            = 0;   // variabile di memorizzazione del movimento servo base sx
int movcorpodx           = 0;   // variabile di memorizzazione del movimento servo corpo dx
int movbasedx            = 0;   // variabile di memorizzazione del movimento servo base dx
int comododicalcolo      = 0;   // zona di calcolo per l'individuazione del movimento maggiore
int comodo1              = 0;   // zona di calcolo per l'individuazione del movimento maggiore
int comodostep           = 0;   // variabile utilizzata nella routine di movimento dei carrelli
int movmag               = 0;   // zona di memorizzazione del movimento maggiore
int incremento           = 0;   // memorizzazione incremento/decremento (+1 o -1) dei gradi dei servo
int elemento             = 0;   // indice assoluto del primo elemento della riga in esame

/* tabella delle mosse degli stepper e dei servo.
* Per gli stepper sono indicati gli impulsi (positivi o negativi a seconda che si voglia far girare
* il perno in senso orario o antiorario) mentre per i servomotori sono indicati i gradi che devono
* essere raggiunti da ogni singolo servo. Il settimo elemento di ogni riga e' il tempo (in
* millisecondi), che il sistema deve attendere prima di eseguire i movimenti indicati nella riga
* successiva. Il tempo massimo di attesa, per ogni riga e' di 30 secondi (30000 millisecondi) per
* via del fatto che ogni variabile di tipo int puo' contenere solo numeri compresi tra -32767 e
* +32767
*
*-----------------------------------------------------------------------------------------------
* stepper and servo moves array.
* columns: left stepper, right stepper, left base servo, left body servo, right base servo, right
* body servo, delay time (in milliseconds).  For the steppers are indicated pulses (positive or
* negative depending on whether you want rotate: clockwise or counterclockwise), while for
* servomotors are indicated grades that must be achieved on each servo. The seventh element of
* each line, is time (in milliseconds) that the system waits before performing the next row
* movements. The maximum waiting time, for each row, is 30 seconds (30000 milliseconds) due to
* the fact that every "int" variable can only contain numbers between -32767 and +32767
*-----------------------------------------------------------------------------------------------
*
*/
int mosse [71]  = {
  99,
//stsx  stdx  corposx     basesx  corpodx    basedx   delay
    0,    0,       00,       30,       0,       107,   10000,   // posizione iniziale
 1000, 1000,       55,       30,      55,       107,   30000,   // posizione di visione
    0,    0,       55,       30,      55,       107,   30000,   // attesa per la visione
    0,    0,       55,       30,      55,       107,   25000,   // attesa per la visione
 -500, -500,       55,      110,      55,       107,      50,   // posizione intermedia
 -500, -500,       60,      110,      60,        25,    5000,   // chiusura faccia a faccia
  500,  500,       60,      110,      60,        25,      50,   // posizione intermedia
  500,  500,       55,       30,      55,       107,   12000,   // posizione di visione
    0,    0,       55,       30,      55,       107,   30000,   // attesa per la visione
-1000,-1000,       00,       30,       0,       107,    5000,   // posizione iniziale
};
//*
//****  routine di individuazione e memorizzazione del movimento maggiore (gradi o impulsi)
//*
//
void verificamaggiore (void)
{
  comodo1 = abs(comododicalcolo);
  if (comodo1 > movmag)
    movmag = comodo1;
}
//
//
void setup()
{
  Serial.begin (9600);
  stdestro.setSpeed(200);     // imposta una media velocita' di rotazione dell'albero
  stsinistro.setSpeed(200);   // dei due motori passo passo: 200 impulsi al secondo, pari a
  // 6,25 giri al secondo dell'albero motore e ad una rotazione di 35 gradi al secondo
  // del perno in uscita
  basesx.attach(7);     // assegna il servo oggetto "basesx" alla porta 7
  basedx.attach(3);     // assegna il servo oggetto "basedx" alla porta 3
```

```
  corposx.attach(6);   // assegna il servo oggetto "corposx" alla porta 6
  corpodx.attach(4);   // assegna il servo oggetto "corpodx" alla porta 4

  // posiziona i servomotori in posizione iniziale  (basi allineate tra loro e corpo parallelo al
terreno)
  basesx.write(30);         // indirizza il perno di basesx alla posizione 50
  precbasesx          = 30;
  basedx.write(107);        // indirizza il perno di basedx alla posizione 107
  precbasedx          = 107;
  corposx.write(0);         // indirizza il perno di corposx alla posizione 00
  preccorposx         = 0;
  corpodx.write(0);         // indirizza il perno di corpodx alla posizione 00
  preccorpodx         = 0;

  /*
 ***** routine di utilizzo della tabella mosse e di azionamento di motori e servomotori  **********
 **********************  moves array: using routine ***************************************
*
  regola di sincronismo nei movimenti: ad ogni incremento dell'indice k corrisponde
  il movimento 1 grado angolare e 10 impulsi agli stepper, sempre che non abbiano gia'
  raggiunto la loro posizione finale
*
* rules: each upgrade of k index means 1 angular degree on servo or 10 pulses to stepper
*/
  for ( i = 1; i <= 10; i++)   //  for di scorrimento degli elementi della tabella mosse
  {
    Serial.print ("indice mosse = ");
    Serial.println  (i);
    //
    // individua l'escursione maggiore tra i movimenti presenti nella riga di tabella in esame
    //
    elemento = ((i * 7) - 6);  // calcola la posizione del primo campo della riga in tabella mosse
    movmag = 0;                // azzera  la variabile di memorizzazione dell'escursione maggiore

    movstsinistro     = (mosse [elemento] / 10);
    comododicalcolo   = movstsinistro;
    verificamaggiore ();
    movstdestro       = (mosse [elemento + 1] / 10);
    comododicalcolo   = movstdestro;
    verificamaggiore ();
    comododicalcolo   = (preccorposx - (mosse [elemento + 2]));  //calcola la differenza tra
// l'attuale posizione  del servo del corpo sinistro e posizione da raggiungere (da tabella mosse)
    verificamaggiore ();
    movcorposx        = mosse [elemento + 2]; // inserisce in movcorposx la posizione che il servo
    //   dovra' raggiungere

    comododicalcolo   = (precbasesx - (mosse [elemento + 3]));
    verificamaggiore ();
    movbasesx         = mosse [elemento + 3]; // inserisce in movbasesx la posizione che il servo
    //   dovra' raggiungere
    comododicalcolo   = (preccorpodx - (mosse [elemento + 4]));
    verificamaggiore ();
    movcorpodx        = mosse [elemento + 4]; // inserisce in movcorpodx la posizione che il servo
    //  dovra' raggiungere

    comododicalcolo   = (precbasedx - (mosse [elemento + 5]));
    verificamaggiore ();
    movbasedx         = mosse [elemento + 5]; // inserisce in movbasedx la posizione che il servo
   //  dovra' raggiungere

    Serial.print ("movamg = ");
    Serial.println  (movmag);
    //
    // ciclo di for per il micro movimento di servomotori e stepper e per traccia di debug
    //
    Serial.print ("posizione iniziale dei servo = ");
    Serial.print  (movstsinistro);
    Serial.print  ("  ");
    Serial.print  (movstdestro);
    Serial.print  ("  ");
    Serial.print  (preccorposx);
    Serial.print  ("  ");
    Serial.print  (precbasesx);
    Serial.print  ("  ");
    Serial.print  (preccorpodx);
    Serial.print  ("  ");
    Serial.println  (precbasedx);
```

```
    for (k = 1; k <= movmag; k++)  //  micromovimenti ripetuti per il numero di volte della maggiore
    // escursione
    {
      // movimento stepper sinistro
      comodostep = 1;              // 10 impulsi per ogni micromovimento
      if (movstsinistro < 0)
        comodostep = -1;      // impulsi negativi (movimento antiorario del perno) se il
      // movimento e' negativo)
      if (movstsinistro == 0)
        comodostep = 0;      // nessun impulso se il carrello ha gia' raggiunto la posizione finale
      movstsinistro = movstsinistro - comodostep;  // aggiorna il movimento residuo da compiere
      stsinistro.step (comodostep * 10);           // muove il carrello

      // movimento stepper destro
      comodostep = 1;
      if (movstdestro < 0)
        comodostep = -1;
      if ( movstdestro == 0)
        comodostep = 0;
      movstdestro = movstdestro - comodostep;
      stdestro.step (comodostep * 10);

      // movimento servo corpo sx
      incremento = 1;                    // 1 grado per ogni micromovimento
      if (preccorposx > movcorposx)      // se i gradi di partenza sono maggiori dei gradi di arrivo
        incremento = -1;                 // movimento negativo di un grado
      if (preccorposx == movcorposx)     // se il servo ha gia' raggiunto la posizione finale
        incremento = 0;                  // movimento di 0 gradi
      preccorposx = preccorposx + incremento;  /* calcola la posizione che deve raggiungere il servo
       (un grado in piu' o in meno della posizione precedente oppure, se gia' arrivato alla meta,
       nessuna variazione)
*/
      corposx.write (preccorposx);     // muove il servomotore

      // movimento servo corpo dx
      incremento = 1;
      if (preccorpodx > movcorpodx)
        incremento = -1;
      if (preccorpodx == movcorpodx)
        incremento = 0;
      preccorpodx = preccorpodx + incremento;
      corpodx.write (preccorpodx);

      // movimento servo base sx
      incremento = 1;
      if (precbasesx > movbasesx)
        incremento = -1;
      if (precbasesx == movbasesx)
        incremento = 0;
      precbasesx = precbasesx + incremento;
      basesx.write (precbasesx);

      // movimento servo base dx
      incremento = 1;
      if (precbasedx > movbasedx)
        incremento = -1;
      if (precbasedx == movbasedx)
        incremento = 0;
      precbasedx = precbasedx + incremento;
      basedx.write (precbasedx);
      delay (50);
    }
    delay (mosse [elemento + 6]); // attende il tempo previsto dal settimo campo di ogni elemento in
    //   tabella mosse
  }
}

void loop()
{
// nessuna istruzione nella zona di loop
// no statements on this section
}
```