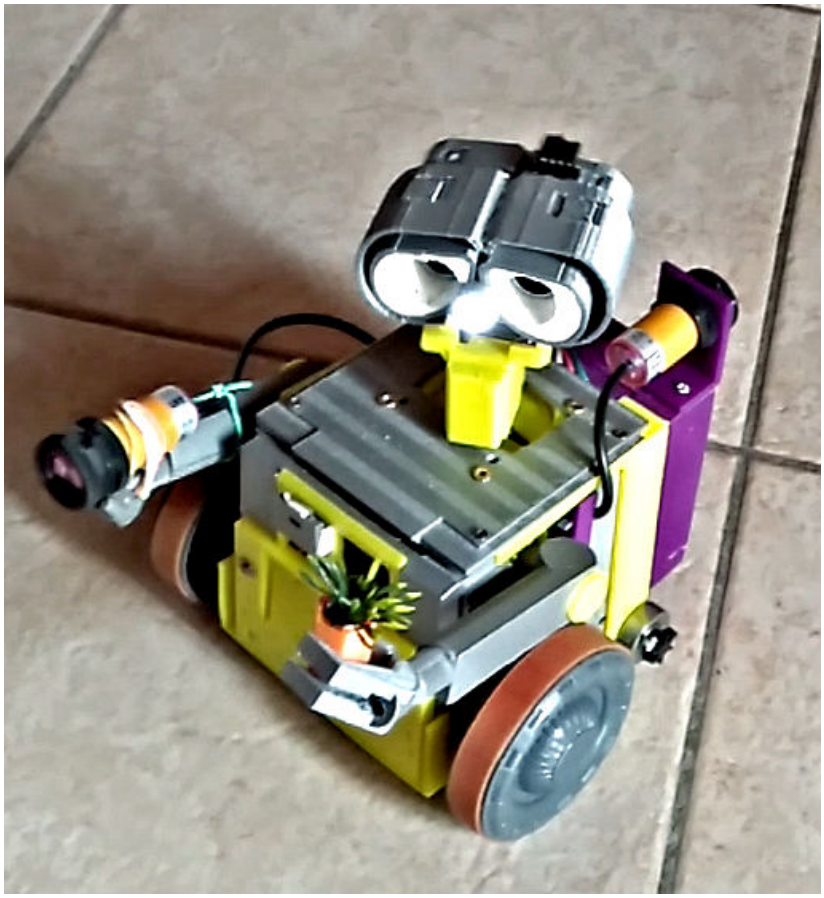
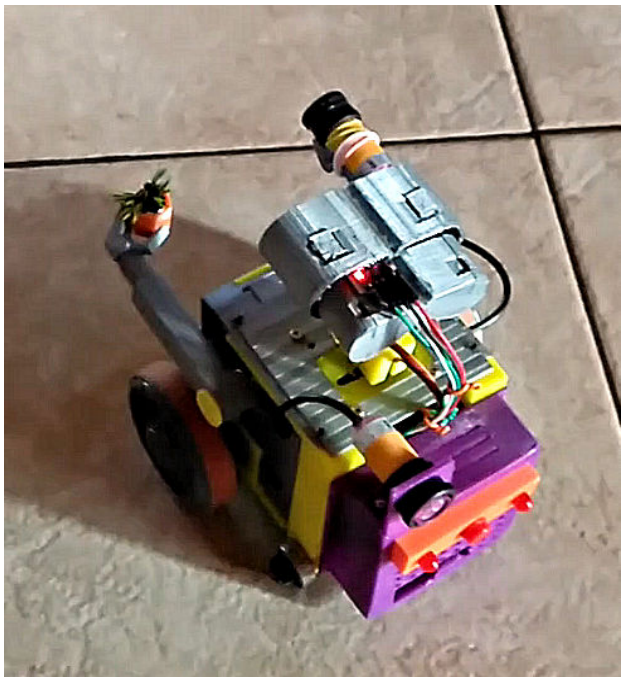
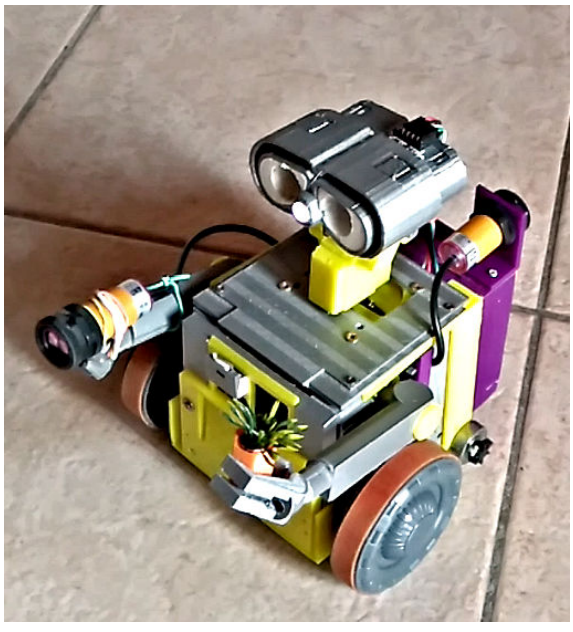


Wall_E

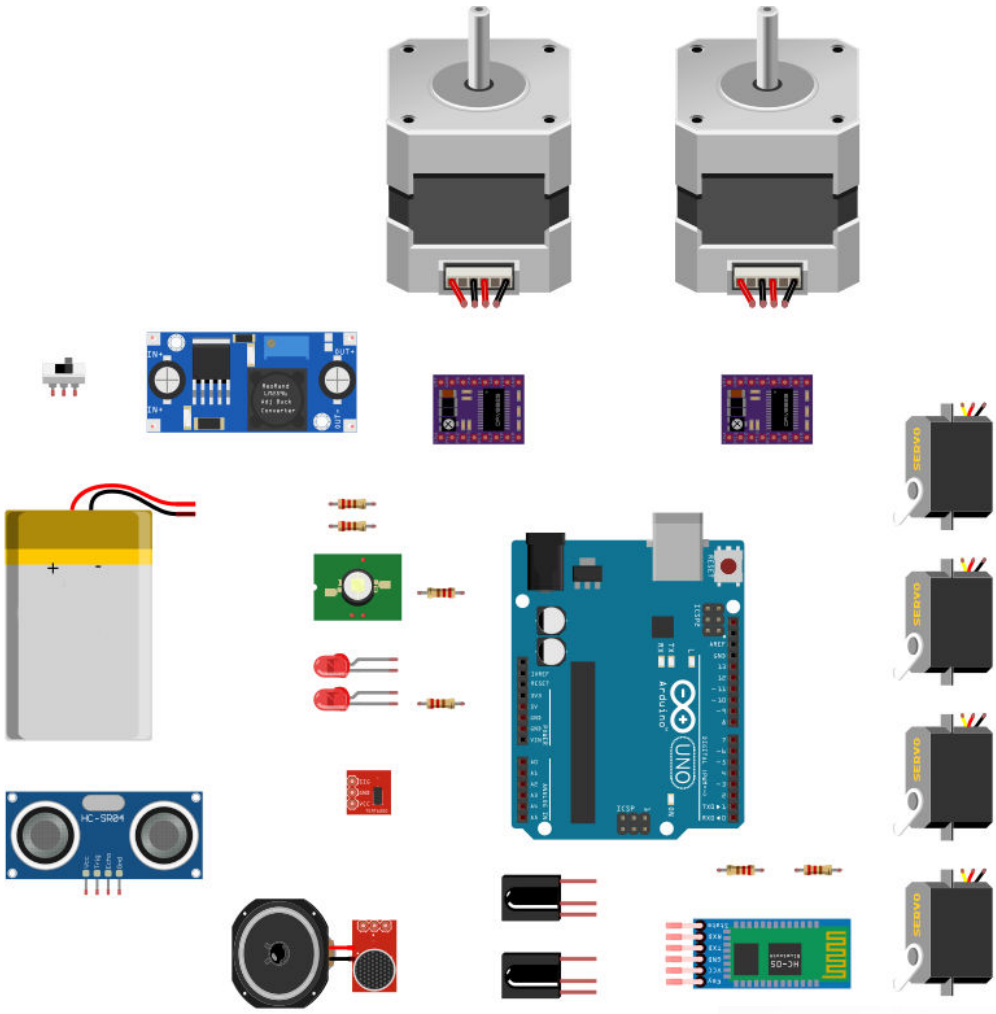




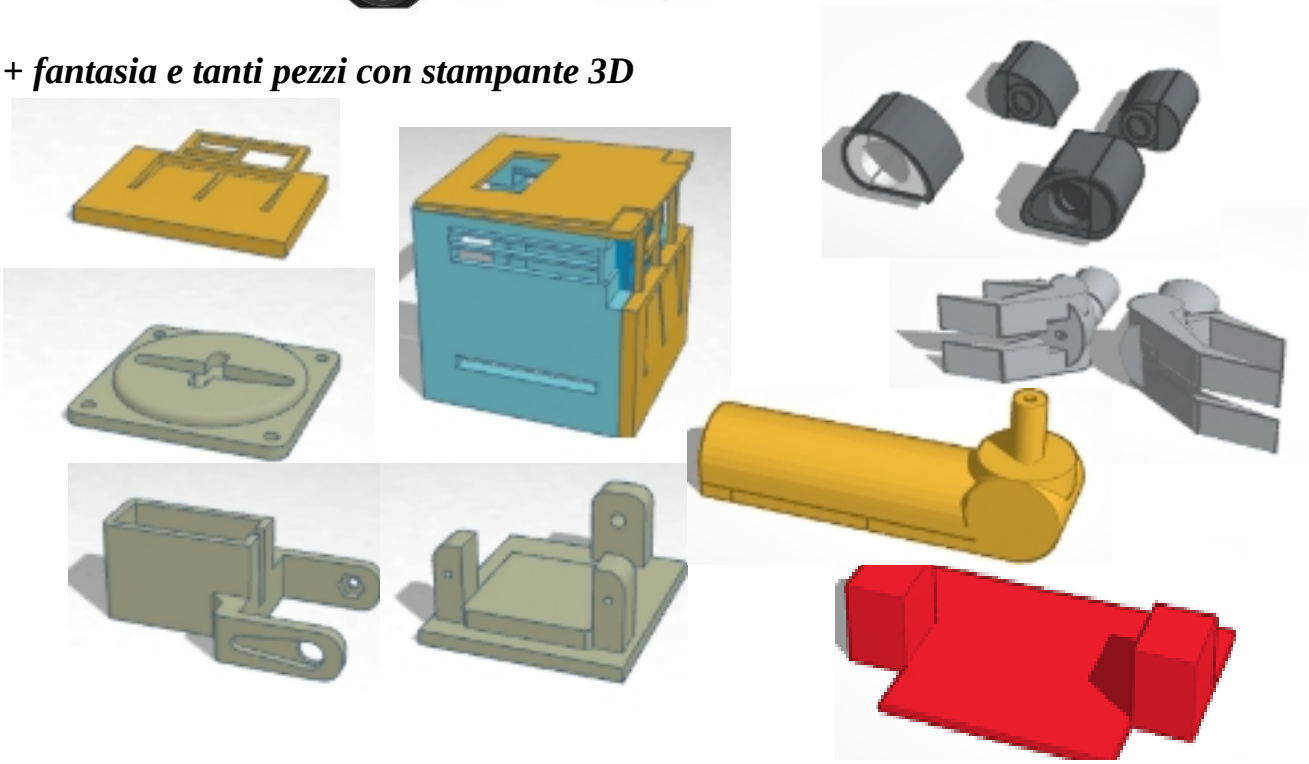
Arduino processor

Arrigo 10-2019

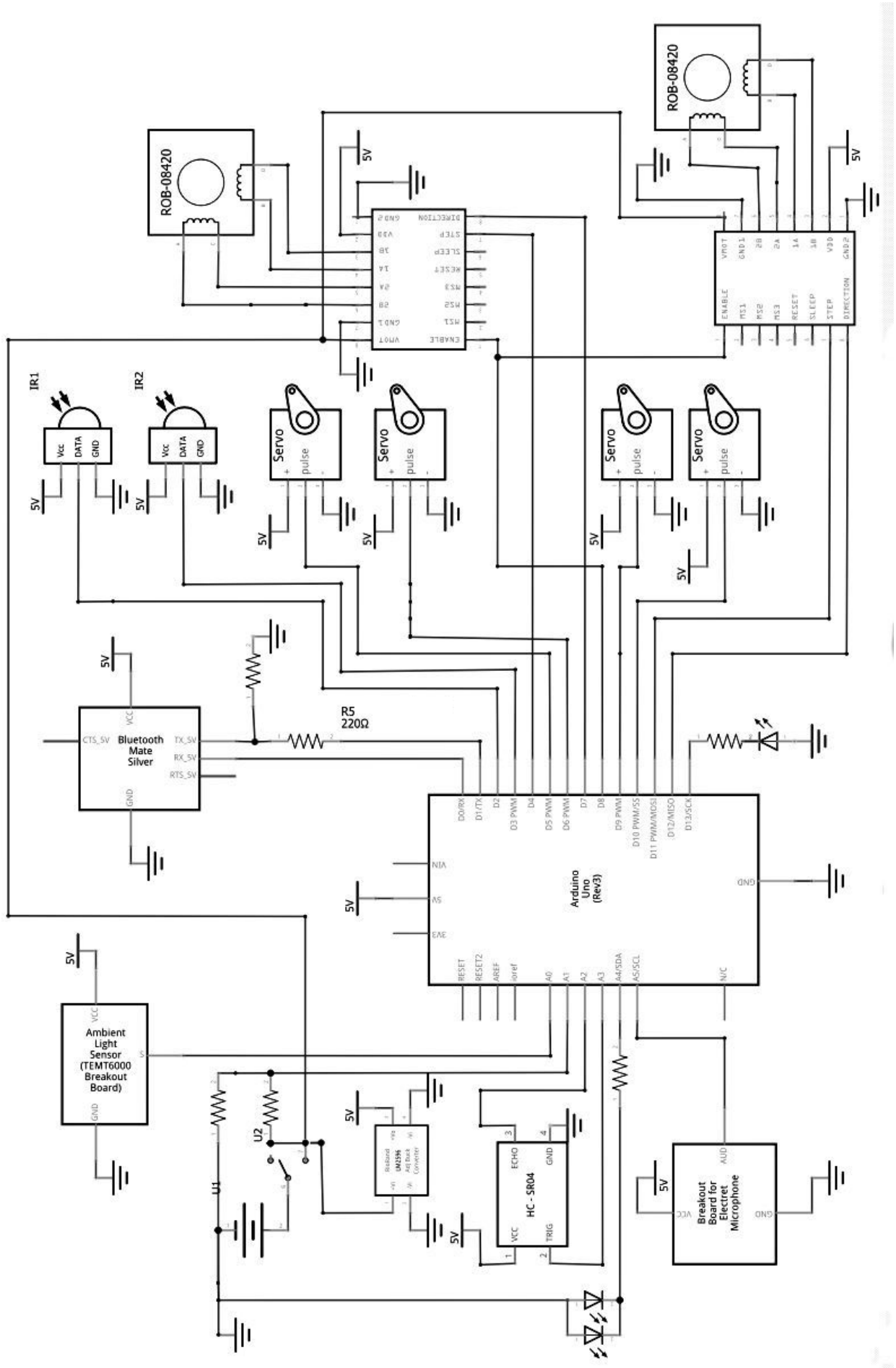
Hardware necessario



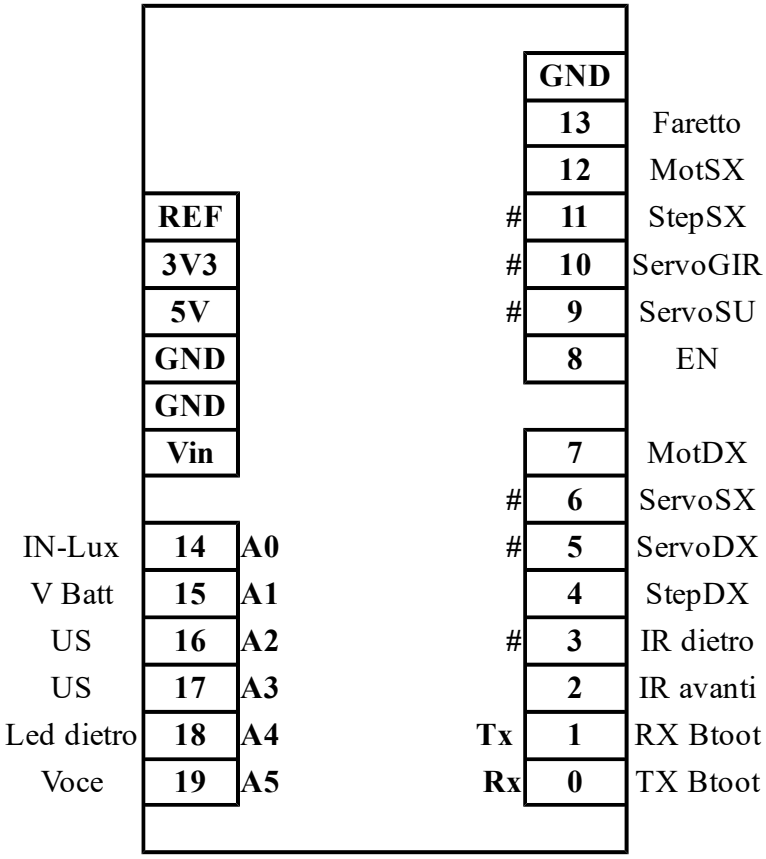
+ fantasia e tanti pezzi con stampante 3D



Questo lo schema

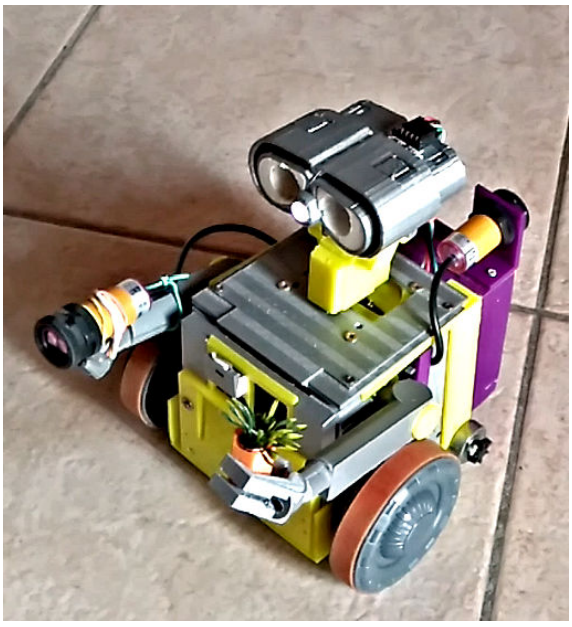


questi i collegamenti specifici su Arduino Uno



Qui un piccolo filmato con Wall_E all'opera

https://www.youtube.com/watch?v=BG_1Wj6bB-Q&feature=youtu.be



seguimi su ***http://fotoarrigo.com*** ***instagram : xarrigo***

Questo il programma (migliorabile)
in blu i commenti, in rosso le stringhe in uscita, per la lettura su smartphone ho usato l'APP Serial Bluetooth Terminal di Kai Morich

```
// Robottino Wall_E

#include "Ultrasonic.h"
#include <Wire.h>
#include <Arduino.h>
#include <OneWire.h>
#include <Servo.h>

# define EN 8         // gestione  motori stepper , attivi LOW , blocco HIGH
# define MotSX 12 // SX direzione motore stepper, avanti LOW , indietro HIGH
# define MotDX 7  // DX direzione motore stepper, avanti LOW , indietro HIGH
# define SX_Step 11 // SX controllo impulsi motore stepper da cui dipende la velocità
# define DX_Step 4  // DX controllo impulsi motore stepper da cui dipende la velocità

const int numStepMotore = 200; //E' il numero di step per una rotazione del motore
//microsecondi tra un impulso e l'altro sul pin STEP
const int vel_ind = 2000;
const int vel_avn = 1000;
const int vel_cur = 3000;
const int vel_avp = 3500;

Ultrasonic ultrasonic(16, 17);   //pin misura US

Servo dxservo;    // crea l'oggetto servo braccio destro
Servo sxservo;     // crea l'oggetto servo braccio sinistro
Servo suservo;    // crea l'oggetto servo movimento verticale testa
Servo girservo;   // crea l'oggetto servo movimento orizzontale testa
int pos1 = 35;     // variable e posizione iniziale del servo dx
int pos2 = 140;   // variable e posizione iniziale del servo sx
int pos3 = 100;   // variable e posizione iniziale del servo su
int pos4 = 73;     // variable e posizione iniziale del servo gir

//========== VARIABILI ==================
int lux;          //valore luminosità convertitore A/D max = 1023
int led = 13;    //pin attivazione faretto
int val = 0;      //valore iniziale misura US
int valdx;       //valore misura US lato destro
int valsx;       //valore misura US lato sinistro
int irav;         //situazione IR avanti
int irrt;         //situazione IR dietro
int a = 0;       //variabile case
int bat;         //valore digitale batteria pin A1
float bat2;     //valore calcolato Volt batteria

//====================***==================
void setup () {
  Serial.begin(9600);    //attivo comunicazione seriale PC-Arduino
  Wire.begin();
  dxservo.attach(5);     // dichiaro il servo collegato al pin 5
  sxservo.attach(6);     // dichiaro il servo collegato al pin 6
  suservo.attach(9);     // dichiaro il servo collegato al pin 10
  girservo.attach(10);  // dichiaro il servo collegato al pin 11

  dxservo.write(pos1);  //metto il dxservo nella posizione iniziale
  sxservo.write(pos2);  //metto il sxservo nella posizione iniziale
  suservo.write(pos3);  //metto il suservo nella posizione iniziale
  girservo.write(pos4); //metto il girservo nella posizione iniziale
```

```
  Serial.println(" ** Robottino Wall_E_Stepper **");
  Serial.println();
  pinMode (MotSX, OUTPUT);    //motore sinistro
  pinMode (MotDX, OUTPUT);    //motore destro
  pinMode (SX_Step, OUTPUT); //step motore sinistro
  pinMode (DX_Step, OUTPUT); //step motore destro
  pinMode (EN, OUTPUT);          //ON-OFF movimento motori
  pinMode (18, OUTPUT);          //pin led posteriori
  pinMode (19, OUTPUT);          //pin attivazione messaggio voce
  digitalWrite (19, LOW);        // voce spenta
  pinMode (13, OUTPUT);          // pin led faretto
  pinMode (10, OUTPUT);          // servo rotazione testa
  pinMode (5, OUTPUT);           // servo braccio dx
  pinMode (6, OUTPUT);           // servo braccio sx
  pinMode (9, OUTPUT);           // servo testa SU-GIU
  pinMode (2, INPUT);            // gestione interrupt retro
  pinMode (3, INPUT);            // gestione interrupt avanti

  digitalWrite (EN, HIGH);   //abilito i motori al movimento
  attachInterrupt(0, blocco, FALLING);   // IR avanti
  attachInterrupt(1, blocco, FALLING);   // IR dietro

  dxservo.write(35);    //sensore avanti in posizione
  sxservo.write(155);   //posizione braccio sx
  delay(1000);
}

//===============****** FINE SETUP ********===============

void loop () {
  attachInterrupt;
  batteria();   //misura tensione di batteria
  LedOnOff();   // verifico necessità accensione faretto
  delay(150);
  irav = digitalRead(2);  //  stato del sensore davanti
  irrt = digitalRead(3);   //  stato del sensore dietro
  delay(50);
  a = 0;
  //==================****** C A S E ********===============

  Serial.println (" Misuro distanza avanti ");
  //determino case se spazio avanti minore di 80cm
  val = ultrasonic.Ranging(CM); delay(800);
  Serial.println (" Analizzo CASE ");
  if ((irav > 0) && (irrt > 0) && (val > 85)) a = 1;  //nessun allarme
  if ((irav < 1) | (val < 80) && (irrt > 0))      a = 2;  //allarme avanti
  if ((irav > 0) && (irrt < 1))                   a = 3; //allarme dietro
  if ((irav < 1) && (irrt < 1))                   a = 4; //allarme avanti e dietro

  //==================== SWITCH ============================
  switch (a) {

    case 1:              // nessun sensore IR attivo spazio > 80cm
      dxservo.write(35); delay(50);
      Serial.println(" CHIAMO EVA ");   // eseguo il messaggio vocale
      digitalWrite (19, HIGH); delay(2500); digitalWrite (19, LOW);
      Serial.println(" MOTORI AVANTI ");
      irav = digitalRead(2); //  stato del sensore davanti
      if (irav < 1) break;
      digitalWrite (EN, LOW);
      digitalWrite(MotSX, LOW); digitalWrite(MotDX, LOW);
      for (int x = 0; x < 4000; x++) {
```

```
      digitalWrite(SX_Step, HIGH); digitalWrite(DX_Step, HIGH);
      delayMicroseconds(vel_avn);
      digitalWrite(SX_Step, LOW); digitalWrite(DX_Step, LOW);
      delayMicroseconds(vel_avn);
     }
     break;
   //=========================================================

   case 2:       // allarme sensore IR avanti o spazio < 80cm
     digitalWrite (EN, LOW);
     Serial.println(" OSTACOLO AVANTI - INDIETRO PIANO");
     Serial.println(" ");
      //due giri indietro piano
     digitalWrite (EN, LOW);
     digitalWrite(MotSX, HIGH); digitalWrite(MotDX, HIGH);
     for (int x = 0; x < numStepMotore; x++) {
      digitalWrite(SX_Step, LOW); digitalWrite(DX_Step, LOW);
      delayMicroseconds(vel_avp);
      digitalWrite(SX_Step, HIGH); digitalWrite(DX_Step, HIGH);
      delayMicroseconds(vel_avp);
     }
     digitalWrite (EN, LOW);
     for (int x = 0; x < numStepMotore; x++) {
      digitalWrite(SX_Step, LOW); digitalWrite(DX_Step, LOW);
      delayMicroseconds(vel_avp);
      digitalWrite(SX_Step, HIGH); digitalWrite(DX_Step, HIGH);
      delayMicroseconds(vel_avp);
     }
     gira();    //misuro dx e sx ; decido verso quale direzione girare
     dxservo.write(35); delay(50); // sensore avanti in posizione
     break;
   //=========================================================

   case 3:              // allarme sensore IR dietro
     digitalWrite (EN, LOW);
     Serial.println(" OSTACOLO DIETRO -- AVANTI PIANO");
     Serial.println(" ");
     digitalWrite (EN, LOW);
     //un giro avanti piano
     digitalWrite(MotSX, LOW); digitalWrite(MotDX, LOW);
     for (int x = 0; x < numStepMotore; x++) {
      digitalWrite(SX_Step, LOW); digitalWrite(DX_Step, LOW);
      delayMicroseconds(vel_avp);
      digitalWrite(SX_Step, HIGH); digitalWrite(DX_Step, HIGH);
      delayMicroseconds(vel_avp);
     }
     gira();  //misuro dx e sx ; decido verso quale direzione girare
     dxservo.write(35);  delay(50); //sensore avanti in posizione
     break;
   //=========================================================

   case 4:            // allarme sensore IR avanti e dietro
     digitalWrite (EN, HIGH);
     Serial.println(" OSTACOLO DIETRO E AVANTI --- BLOCCO MOTORI");
     Serial.println(" ");
     lamp();   // lampeggio per allarme
     break;
  }
}
//===============****** FINE LOOP ********===============
```

```
//==================*** B L O C C O ***================
void blocco() {
  // blocco immediato entrambi i motori
  Serial.println(" BLOCCO MOTORI ");
  Serial.println(" ");
  digitalWrite (EN, HIGH);
  delay(100);
}

//==================****** B A T T E R I A ********=========
void batteria() {    //controllo stato batteria
  delay(50);
  bat = analogRead(1); delay(50);
  bat2 = bat * 0.0068;
  while (bat2 < 6.90)   //loop batteria bassa con allarme ottico
  {
    digitalWrite (EN, HIGH);
    Serial.print(" BATTERIA BASSA "); Serial.print(bat2); Serial.println(" V");
    Serial.println(" ");
    digitalWrite (led, LOW);
    delay(500);
    digitalWrite (led, HIGH);
    delay(500);
    digitalWrite (led, LOW);
    delay(500);
    digitalWrite (led, HIGH);
    delay(500);
    bat = analogRead(1); delay(50);
    bat2 = bat * 0.0068;
  }
  Serial.print(" BATTERIA OK    "); Serial.print(bat2); Serial.print(" V");
  Serial.println(" ");
}

//==============****** F A R E T T O ********===============
void LedOnOff() {
  //accendo o spengo faretto in relazione luminosità ambiente
  lux = analogRead(0); delay(50);
  if (lux < 40) {
    digitalWrite (led, LOW); Serial.print(" Poca luce, accendo il faro  ");
    Serial.println(" ");
  }
  else {
    digitalWrite (led, HIGH); Serial.print(" Luce OK, il faro non serve  ");
    Serial.println(" ");
  }
}

//================**** L A M P E G G I O  ****===============
void lamp() {
    //lampeggio per cambio direzione o allarme grave
    Serial.println(" BLINK-BLINK-OCIOO"); Serial.println(" ");
  int i;
  for (i = 0; i < 4; i++) {
    digitalWrite (13, HIGH);
    digitalWrite (18, HIGH);
    delay(300);
    digitalWrite (13, LOW);
    digitalWrite (18, LOW);delay(300);
      }
}

//==================****** G I R A  ********================
```

```
void gira() {
  //determino se è possibile girare e in che direzione
  Serial.print(" Misuro spazio a destra ");  Serial.println(" ");
  delay(50);    // metto il sensore avanti in posizione riposo
  for (pos4 = 73; pos4 > 2; pos4 -= 1) // gira testa a destra
  {
    girservo.write(pos4);
    delay(10); //ritardo per permettere il posizionamento
  }
  delay(300);
  valdx = ultrasonic.Ranging(CM); delay(800);
  Serial.print(" Misuro spazio a sinistra ");  Serial.println(" ");
  for (pos4 = 15; pos4 < 170; pos4 += 1)  // gira testa a  sinistra
  {
    girservo.write(pos4);
    delay(10);  //ritardo per permettere il posizionamento
  }
  delay(300);
  valsx = ultrasonic.Ranging(CM); delay(800);
  for (pos4 = 160; pos4 > 73; pos4 -= 1) // riposiziona testa al centro
  {
    girservo.write(pos4);
    delay(10);  //ritardo per permettere il posizionamento
  }

  //determino dove girare
  if (valdx > valsx) {      // giro a destra
    Serial.println(" GIRO a DESTRA ");
    lamp();         //lampeggio per cambio direzione
    digitalWrite (EN, LOW);
    digitalWrite(MotSX, HIGH); digitalWrite(MotDX, LOW);
    for (int x = 0; x < 100; x++) {
      digitalWrite(SX_Step, HIGH); digitalWrite(DX_Step, HIGH);
      delayMicroseconds(vel_cur);
      digitalWrite(SX_Step, LOW); digitalWrite(DX_Step, LOW);
      delayMicroseconds(vel_cur);
    }
    digitalWrite (EN, HIGH);
    delay(100);
  }

  else {              // giro a sinistra
    Serial.println(" GIRO a SINISTRA ");
    lamp();       //lampeggio per cambio direzione
    digitalWrite (EN, LOW);
    digitalWrite(MotSX, LOW); digitalWrite(MotDX, HIGH);
    for (int x = 0; x < 150; x++) {
      digitalWrite(SX_Step, HIGH); digitalWrite(DX_Step, HIGH);
      delayMicroseconds(vel_cur);
      digitalWrite(SX_Step, LOW); digitalWrite(DX_Step, LOW);
      delayMicroseconds(vel_cur);
    }
    digitalWrite (EN, HIGH);
    delay(100);
  }
}
//====********======= F I N E  PROGRAMMA =======********====
```