```
//Hexapod Master 140B Arduino Nano Old Boot Loader
#include <Servo.h>
#include <Wire.h>

// Definiamo gli oggetti che controllano i servomotori
Servo myservo[3][3];

// Definiamo i piedini ai quali sono collegati i servomotori
byte myservopins[3][3]={2,3,4,5,6,7,8,9,10};

// Indice Movimento e posizione
int indiceMovimento=0,indicePosizione=0;

// Posizione attuale
int pos[6][3];

// Posizione da raggiungere
int goToPos[6][3];

// Piedini Tasti
byte pinTasti[4]={14,15,16,17};

//Posizione Default
const byte zampeDefault[6][3]={{90,50,110},{90,50,110},{90,50,110},{90,50,110},{90,50,110},{90,50,110}};

// posizioneZampa: avanti-centro-indietro (Primo motore -20°,0°,+20°) & su-giù (Secondo e terzo motore
+30°-0°,-30° +60°)
  const int positioneZampa[6][3]={{-20,30,0},{0,30,0},{20,30,0},{-20,-30,-60},{0,-30,-60},{20,-30,-60}};

/* Sono memorizzati i valori relativi ai servomotori per 10 posizioni delle zampe dell'hexapod:
(0)Fermo a zampe alzate  (1) In piedi  (0,1) Preparazione movimento  (1,0) Fine movimento (2,3,4,5)
movimento in avanti  (5,4,3,2) Movimento indietro  (6,7,8,9) Gira antiorario  (9,8,7,6) Gira orario*/
const byte movimentoZampe[10][6]={{1,1,1,1,1,1},{4,4,4,4,4,4},{0,5,0,3,2,3},{3,5,3,3,5,3},{5,0,5,2,3,2},-
{5,3,5,5,3,5},{0,5,0,5,0,5},{3,5,3,5,3,5},{5,0,5,0,5,0},{5,3,5,3,5,3}};

// Stato piedini A0 A1
boolean pinA0=false,pinA1=false;

// Variabili controllo fine movimento
boolean fineMovimento=false,passoEseguito=false;;

// Stato Movimento
byte statoMovimento=0,statoMovimentoBack=0; // 0:Fermo/1:Avanti/2:Finisci Movimento/3:Indietro/4:Ruota
Antiorario/5:Ruota Orario

void setup() {
  byte i,j;
  // Piedini tasti come ingressi
  for(i=0;i<4;i++) pinMode(pinTasti[14+i],INPUT_PULLUP);

  //inizializziamo la trasmissione i2c come Master
  Wire.begin();

  // Definisci velocità di trasmissione
  Serial.begin(9600);

  // Indichiamo i piedini ai quali gli oggetti che controllano i servomotori fanno riferimento
  for(i=0;i<3;i++) for(j=0;j<3;j++) myservo[i][j].attach(myservopins[i][j]);

  // Carichiamo la posizione iniziale
  for(i=0;i<6;i++) for(j=0;j<3;j++) goToPos[i][j]=zampeDefault[i][j];

  delay(1000);
}

// Raggiungi la posizione per i servomotori della scheda master
void raggiungiPosizione(){
  byte i,j;
  for(i=0;i<3;i++) for(j=0;j<3;j++)
  {
    pos[i][j]=myservo[i][j].read();
    if(pos[i][j]!=goToPos[i][j]) {
      myservo[i][j].write(goToPos[i][j]);
    }
```

```
  }

}

byte leggiIngressi(){
  byte i,j,n,cont3;
  boolean numero[6];
  boolean x[4][4];
for(i=0;i<4;i++){
    pinMode(pinTasti[i],OUTPUT);
    digitalWrite(pinTasti[i],LOW);
    for(j=0;j<4;j++){
      if((i!=j)&&(digitalRead(pinTasti[j])==0)) {x[i][j]=true;} else {x[i][j]=false;}
    }
    pinMode(pinTasti[i],INPUT_PULLUP);
  }
  numero[0]=x[0][1]&x[1][0];
  numero[1]=x[0][2]&x[2][0];
  numero[2]=x[0][3]&x[3][0];
  numero[3]=x[1][3]&x[3][1];
  numero[4]=x[2][3]&x[3][2];
  numero[5]=x[1][2]&x[2][1];
  n=255;
  cont3=0;
  for(i=0;i<6;i++)if(numero[i]){
    n=i;
    cont3++;
  }
  if(cont3==1) {return n;} else {return 255;}
}

// Visualizza posizione motori
void visualizzaPosizioneMotori(){
  byte i,j;
  Serial.print("posizione: ");
  Serial.println(indicePosizione);
  for(i=0;i<6;i++){
    Serial.print("\t Zampa ");
    Serial.print(i+1);
    Serial.print(":(");
    Serial.print(pos[i][0]);
    Serial.print("\t");
    Serial.print(pos[i][1]);
    Serial.print("\t");
    Serial.print(pos[i][2]);
    Serial.print(") ");
    if(i==2) Serial.println(" ");
    if(i==5)
    {
      Serial.println(" ");
      Serial.println(" ");
    }
  }
}

// Comunicazione dati tra Master e Slave
void comunicazione(byte motore){
  byte x;
  // Inizia trasmissione
  Wire.beginTransmission(0x04);

  //Invio un byte
  x=motore+(goToPos[3+motore/3][motore%3]/10)*10;
  Wire.write(x);

  // Fine trasmissione
  Wire.endTransmission();

  // Ritardo
  delay(1);

  // Richiedo un byte allo slave che ha indirizzo 0x04
  Wire.requestFrom(0x04, 1);
```

```
  // Attendo la disponibilità di dati sul bus i2c
  while(Wire.available())
  {
    // Quando è presente un dato avvia la lettura
    pos[3+motore/3][motore%3] = Wire.read();
  }
}

// Seleziona posizione
void selezionaPosizione(){
  // Controllo di fine movimento
  if((statoMovimentoBack!=0)&&(statoMovimento!=statoMovimentoBack))fineMovimento=true;

  if((fineMovimento)&&(passoEseguito))
  {
    statoMovimento=2;
    statoMovimentoBack=2;
    indicePosizione=2;
    fineMovimento=false;
  }

  // Seleziona posizione in base al movimento impostato
  switch(statoMovimento){
    // Hexapod fermo a zampe alzate
    case 0:
      indicePosizione=0;
    break;

    // Hexapod avanti
    case 1:
      indicePosizione++;
      if(indicePosizione>5) {
        indicePosizione=2;
        passoEseguito=true;
      }
      else passoEseguito=false;
      if(statoMovimentoBack != statoMovimento) indicePosizione=1;
      statoMovimentoBack=statoMovimento;
    break;

    // Finisci movimento
    case 2:
      indicePosizione--;
      if(indicePosizione<0)
      {
        statoMovimento=0;
        statoMovimentoBack=0;
        indicePosizione=0;
      }
    break;

    // Hexapod indietro
    case 3:
      indicePosizione--;
      if(indicePosizione<2){
        indicePosizione=5;
        passoEseguito=true;
      }
      if(statoMovimentoBack != statoMovimento) indicePosizione=5;
      statoMovimentoBack=statoMovimento;
    break;

    // Hexapod ruota antiorario
    case 4:
      indicePosizione--;
      if(indicePosizione<6){
        indicePosizione=9;
        passoEseguito=true;
      }
      if(statoMovimentoBack != statoMovimento) indicePosizione=9;
      statoMovimentoBack=statoMovimento;
    break;

    // Hexapod ruota orario
```

```
    case 5:
      indicePosizione++;
      if(indicePosizione>9){
        indicePosizione=6;
        passoEseguito=true;
      }
      if(statoMovimentoBack != statoMovimento) indicePosizione=6;
      statoMovimentoBack=statoMovimento;
    break;

    default:
      indicePosizione=1;
    break;
  }
}

void loop() {
  byte i,j;
  int x;
  // Visualizza posizione motori
  visualizzaPosizioneMotori();

  // Leggi ingressi e seleziona movimento
  statoMovimento = leggiIngressi();

  // Seleziona posizione in base al movimento impostato
  selezionaPosizione();

  // Carica la posizione dei servomotori nella matrice goToPos[6][3]
  for(i=0;i<6;i++)for(j=0;j<3;j++)goToPos[i][j]=zampeDefault[i][j]-
+positioneZampa[movimentoZampe[indicePosizione][i]][j];

  // Avvia comunicazione dati tra Master e Slave e manda i dati dei motori allo slave
  for(i=0;i<9;i++) comunicazione(i);

  // Raggiungi la posizione per i servomotori della scheda master
  raggiungiPosizione();

  // Ritardo per ogni ciclo
  delay(50);
}
```