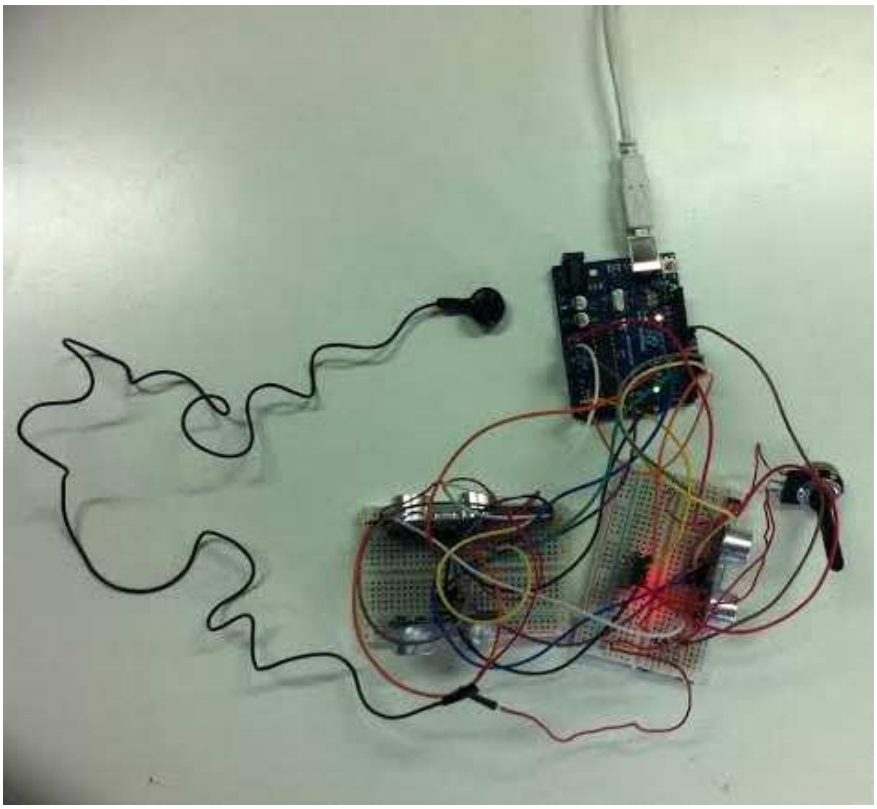
Classe: 4^Q A.S 14-15

I.I.S.S. FOSSATI – DA PASSANO
LA SPEZIA

Relazione di Telecomunicazioni

Supporto per non-vedenti con Arduino



Componenti del gruppo:

- Alexander Cozzani: riassemblatore e programmatore

- Adelmo Brunelli: saldatore e collaudatore

- Laura Pernice: programmatrice ed editrice

- Alessio D'Addario: programmatore, assemblatore ed editore

- Sharon Paoletti: programmatrice ed editrice

- Davide Bennati: programmatore ed editore

Abstract or Summary

Questo progetto fin dall'inizio si pone come obiettivo quello di creare un supporto per non vedenti con l'aiuto di uno strumento di tecnologia economica come Arduino e relativi sensori e sistemi.

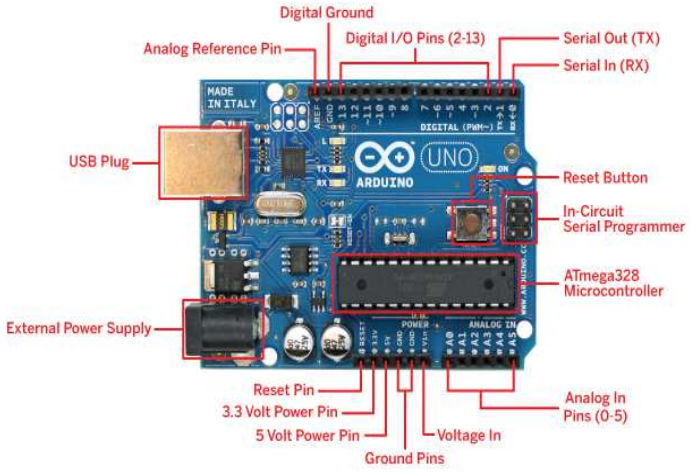
This project from the start it arises as target of create a support for visualy impared with help of a economic technology instrument as Arduino and related sensory and system.

Introduzione

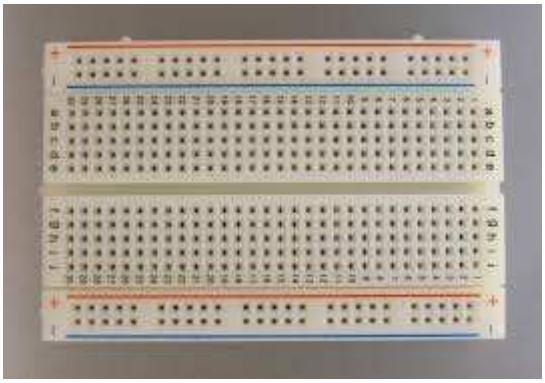
Questo progetto si è posto fin dall'inizio l'obiettivo di creare un supporto per persone non-vedenti con l'aiuto di uno strumento tecnologico ed economico come Arduino, utilizzando anche relativi sensori.

Materiali Hardware utilizzati

- MicroController Arduino Uno (ATMega2548)
  E' un hardware di basso costo che possiede le stesse proprietà di un computer, quindi possiamo classificarlo come un "piccolo computer".



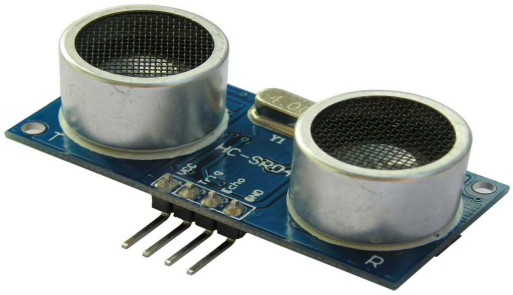
- BreadBoard
  E' un supporto utilizzato principalmente nel campo dell'elettronica per la realizzazione di circuiti elettrici.



- Cavi di Interconnessione
  Cavi, realizzati di rame, utilizzati per connettere i vari dispositivi sulla breadboard

- Sensore HC-04

  E' un chip che segnala le distanze dal primo ostacolo, rappresentando i dati captati su una videata che comprende i tre assi cartesiani. Utilizza principalmente segnali ultrasuoni.



- MAG3110

  Questo componente può essere usato in associazione con 3 assi che produce l'effetto di una bussola elettronica indipendente che fornisce accurate informazioni sulla posizione attuale.

- Transistor BC107

  E' un transistor bipolare di tipo BJT , è un semiconduttore usato per amplificare il segnale elettrico.

- Magnete

E’ un pezzo di ferrite utilizzato per generare un campo magnetico attorno ad esso.

- Cuffia auricolare

  E’ un dispositivo posto dentro il padiglione auricolare che serve per ascoltare i segnali audio emanati dal nostro programma.
- Speaker

  E’ un dispositivo connesso al circuito che emana in output un segnale audio che, nel nostro caso, avvisa il non vedente di un pericolo imminente.

Componenti software

- MicroCap

  E’ un editor di circuiti elettronici e un simulatore analogico/digitale che permette la costruzione del disegno e la simulazione del circuito.

- Java(Processing)

  E’ un linguaggio di programmazione open-source che consente di sviluppare diverse applicazioni, inoltre eredita la sintassi ed i comandi della programmazione ad oggetti.

Procedimento

*- **Step 1** : collegare i sensori HC-04 inseriti nella breadboard all'arduino: il primo pin a terra (gnd),il secondo (trig) e terzo pin (echo) rispettivamente a due pin digitali (secondo l'inizializzazione scritta nel programma), l'ultimo pin a +5v.*
Il paziente attraverso questi sensori potrà rilevare la posizione di un ostacolo grazie al suono emesso dalla cuffia, questo suono aumenterà di intensità quando l'ostacolo sarà più vicino.

*- **Step 2** : collegare il transistor bc107 inserito nella breadboard: il primo prin a terra(gnd),il secondo pin al potenziometro da una parte e dall'altra al pin dell'arduino, l'ultimo pin a +5v.*
A sua volta il potenziometro sarà collegato a terra (gnd) e alla cuffia la quale è collegata a terra (gnd).

*- **Step 3** : collegare il mag3110 inserito nella breadboard all'arduino: il primo pin a +3,3v (vcc), il secondo pin a terra (gnd), il terzo pin (sda) al pin A4 (seriale), il quarto pin (scl) al pin A5 (seriale).*

Quest'ultimo cambierà i suoi assi attraverso l'utilizzo di magneti per evitare in modo ancora più preciso gli ostacoli.
In seguito, attraverso il programma, verrà inizializzato il pin di uno speaker collegato all'arduino e a terra (gnd) questo agirà in sicrono con l'avvicinamento dei magneti: più saranno vicini più lo speaker suonerà con più intensità.

*- **Step 4** : creare il programma con MicroC, inizializzando i diversi ingressi (a seconda di dove sono stati inseriti nell'arduino).*

*- **Step 5** : verificare che non ci siano errori compilando il programma ed eseguendolo ; in questo modo sarà inserito nell'arduino (collegato al computer tramite cavo USB).*

*- **Step 6** : avviare il monitor seriale e verificare se i componenti producono il risultato voluto in output.*

*- **Step 7** : avviare processing, inserendo il programma apposito. Dopo averlo compilato ed eseguito, verificare che visualizzi i tre assi.*
Avvicinando i magneti al mag3110 a seconda di dove si trovano si noterà che in output aumenterà l'intensità dei puntini su schermo.

*- **Step 8** : Inserire il tutto in una scatola per facilitare la presa al non vedente.*

*- **Step 9:** per evitare di utilizzare il computer, collegare l'arduino ad una batteria utilizzando il relativo cavo alternativo, successivamente collegando la shield Bluetooth: il primo pin (rxd)*

al pin tx->1, il secondo pin (txd) al pin rx->0, il terzo pin a terra (gnd), l'ultimo pin (vcc) a +5v.
Utilizzando questa shield potremmo visualizzare il monitor seriale e a sua volta agire su tutti i componenti tramite un dispositivo esterno (cellulare,pc...).

Programma in MicroC

```
#include <MAG3110.h>
#include <I2Cdev.h>
#include <Wire.h>

#define MAGM_ADDR  MAG3110_I2C_ADDRESS
#define MAG_ADDR MAGM_ADDR

int triggerPort = 5;
int echoPort = 3;
int triggerPort2 = 8;
int echoPort2 = 7;
int cicalino = 9;
int cicalino2 = 2;
int randNumber;
unsigned long time;
unsigned long lampeggio_time;
unsigned long pausa_time;
void setup() {
pinMode( triggerPort, OUTPUT );
pinMode( echoPort, INPUT );
pinMode( triggerPort2, OUTPUT );
pinMode( echoPort2, INPUT );
pinMode( cicalino, OUTPUT );
pinMode( cicalino2, OUTPUT );

Serial.begin( 9600 );
Wire.begin();
Serial.begin(9600);
delay(100);
Serial.println(mag3110_detect(MAGM_ADDR) ? "Sensore Mag3110 assente..." :  "Sensore Mag3110 rilevato...");
Serial.println("\nOperazioni eseguibili:\n\nPremere a per l'ultrasuono 1 (frontale)\nPremere b per l'ultrasuono 2 (laterale o posteriore)");
Serial.println("Premere m per visualizzare i valori del magnetometro\n\nSelezionare la scelta: ");
mag3110_basic_setup(MAGM_ADDR);
```

```
}
void cicloMag3110(){
delay(500);
  int16_t x, y, z;
  uint8_t st_reg;
  int rc = mag3110_read_all(MAGM_ADDR, &x, &y, &z, &st_reg);
  Serial.print("x= ");
  Serial.print(x);
  Serial.print(", y= ");
  Serial.print(y);
  Serial.print(", z= ");
  Serial.print(z);
  Serial.println("");
  if(x<-600||x>1000)
  {
    digitalWrite(cicalino2, HIGH);
    delay(1000);
    digitalWrite(cicalino2, LOW);
    delay(10);
  }

  //Precisazione sulla rilevazione dei magneti (ulteriore)
   /* if(y<-600||y>1000)
  {
    digitalWrite(cicalino2, HIGH);
    delay(1000);
    digitalWrite(cicalino2, LOW);
    delay(10);
  }
    if(z<-600||z>1000)
  {
    digitalWrite(cicalino2, HIGH);
    delay(1000);
    digitalWrite(cicalino2, LOW);
    delay(10);
  }
 */
}

#define MAG3110_ALL_AXES_READY ((uint8_t)((1 << MAG3110_ZDR) | (1 << MAG3110_YDR) | (1
<< MAG3110_XDR)))

int mag3110_read_all(uint8_t i2c_addr, int16_t* x, int16_t* y, int16_t* z, uint8_t* st_reg_out)
{
  uint8_t st_reg;
  do
  {
    Wire.beginTransmission(i2c_addr);
    Wire.write((uint8_t)MAG3110_DR_STATUS);
    Wire.endTransmission();
```

```
    Wire.requestFrom(i2c_addr, (uint8_t)1);
    st_reg = Wire.read();
  } while (Wire.available() && ((st_reg & MAG3110_ALL_AXES_READY) !=
MAG3110_ALL_AXES_READY));//TODO timeout

  if ( st_reg_out != NULL )
  {
    *st_reg_out = st_reg;
  }

  Wire.beginTransmission(i2c_addr);
  Wire.write((uint8_t)MAG3110_OUT_X_MSB);
  Wire.endTransmission();
  Wire.requestFrom(i2c_addr, (uint8_t)6);

  uint8_t mem[6];
  for (int i = 0; (i < 6) && Wire.available(); i++)
  {
    mem[i^1] = Wire.read();
//    Serial.print("0x");
//    Serial.print(mem[i], HEX);
//    Serial.print(", ");
  }
//  Serial.println("");
  *x = ((int16_t*)mem)[0];
  *y = ((int16_t*)mem)[1];
  *z = ((int16_t*)mem)[2];
  return 0;
}

int mag3110_detect(uint8_t i2c_addr)
{
  Wire.beginTransmission(i2c_addr);
  Wire.write((uint8_t)MAG3110_WHO_AM_I);
  Wire.endTransmission();
  Wire.requestFrom((uint8_t)i2c_addr, (uint8_t)1);
  if (Wire.available())
  {
    uint8_t hello = Wire.read();
    return hello != MAG3110_WHO_AM_I_VALUE;
  }
  return -1;
}

int mag3110_basic_setup(uint8_t i2c_addr)
{
  Wire.beginTransmission(i2c_addr);
  Wire.write((uint8_t)MAG3110_CTRL_REG2);
  Wire.write((uint8_t)(1 << MAG3110_AUTO_MRST_EN));
  Wire.endTransmission();
```

```
  delay(15);//??

  Wire.beginTransmission(i2c_addr);
  Wire.write((uint8_t)MAG3110_CTRL_REG1);
  Wire.write((uint8_t)(1 << MAG3110_AC));
  Wire.endTransmission();
}


void ciclo1(){
Serial.println("----------------------------");
Serial.println("Ultrasuoni 1 (Frontale)");

digitalWrite( triggerPort, LOW );
digitalWrite( triggerPort, HIGH );
delayMicroseconds( 10 );
digitalWrite( triggerPort, LOW );
long duration = pulseIn( echoPort, HIGH );
long r = 0.034 * duration / 2;
Serial.print( "Durata: " );
Serial.print( duration );
Serial.print( " , " );
Serial.print( "Distanza: " );
if( duration > 38000 ) Serial.println( "Fuori portata");
else { Serial.print( r ); Serial.println( "cm" );}
if( r > 3 && r <= 200){
delay(r*10);
digitalWrite(cicalino, HIGH);
delay(r*10); }
if( r <= 3){
digitalWrite(cicalino, HIGH);
delay(1000);}
digitalWrite(cicalino, LOW);
delay(10);
}
void ciclo2(){
Serial.println("----------------------------");
Serial.println("Ultrasuoni 2");

digitalWrite( triggerPort2, LOW );
digitalWrite( triggerPort2, HIGH );
delayMicroseconds( 10 );
digitalWrite( triggerPort2, LOW );
long duration2 = pulseIn( echoPort2, HIGH );
long r2 = 0.034 * duration2 / 2;
Serial.print( "Durata: " );
Serial.print( duration2 );
Serial.print( " , " );
Serial.print( "Distanza: " );
```

```
if( duration2 > 38000 ) Serial.println( "Fuori portata");
else { Serial.print( r2 ); Serial.println( "cm" );}
if( r2 > 3 && r2 <= 200){
delay(r2*10);
digitalWrite(cicalino, HIGH);
delay(r2*10); }
if( r2 <= 3){
digitalWrite(cicalino, HIGH);
delay(1000);}
digitalWrite(cicalino, LOW);
delay(10);

}
void loop() {

char tastiera;
tastiera = Serial.read();
switch (tastiera){
case 'a':
randomSeed(millis());
randNumber = random(200000,400000);
for(int i=0;i<randNumber;i++)
{
ciclo1();
}
break;
case 'b':{
randomSeed(millis());
randNumber = random(200000,400000);
for(int i=0;i<randNumber;i++)
{
ciclo2();
}

break;
case 'm':
randomSeed(millis());
randNumber = random(200000,400000);
for(int i=0;i<randNumber;i++)
{
cicloMag3110();
}
break;
 Serial.flush();
}
} }
```

Programma in Processing

Per vedere l'intensità del campo magnetico del magnete nei diversi tre assi, per visualizzare la sua posizione.

```
import processing.opengl.*;
import processing.serial.*;
float xOffs =  -276.0;
float yOffs = 73.0;
float zOffs = 1251.0;
int bgcolor;
Serial myPort;
int[] serialInArray = new int[6];
int serialCount = 0;
int xpos, ypos, zpos, tpos;
boolean firstContact = false;
boolean DrawAxes;
boolean  BoundariesInitialized;
boolean SerialOnline;
int xMax,xMin,yMax,yMin,zMax,zMin;

int i;

boolean newPoint;
float newX, newY, newZ, newT;

int FGcolorVal;
int Dummy;


color FgColorFunc(int Dummy) {

int rComp = 150;
int gComp = 150;
int bComp = 150;

FGcolorVal += 3;

if (FGcolorVal >= 768)
```

```
  FGcolorVal = 0;

if (FGcolorVal < 256)
{
rComp = 255 - FGcolorVal;
gComp = FGcolorVal;
}
else if (FGcolorVal < 512)
{
gComp = 255 - (FGcolorVal - 256);
bComp = (FGcolorVal - 256);
}
else if (FGcolorVal < 768)
{
bComp = 255 - (FGcolorVal - 512);
rComp = (FGcolorVal - 512);
}
color colortemp = color(rComp, gComp, bComp);

return colortemp;
}



void setup() {
 size(800, 600, OPENGL);
  lights();
  noStroke();
  xpos = width/2;
  ypos = height/2;
int PortCount = Serial.list().length;
  int PortNumber = -1;
  String str1 = "COM5";
  String str2;

  int j = 0;
  while (j < PortCount){
    str2 = Serial.list()[j].substring(0,4);
    if(str1.equals(str2) == true)
```

```
      PortNumber = j;
    j++;
  }
SerialOnline = false;

  if (PortNumber >= 0)
  {
    String portName = Serial.list()[PortNumber];
    myPort = new Serial(this, portName, 57600);
    myPort.buffer(1);
    myPort.clear();
    println("Serial port "+Serial.list()[PortNumber]+" found and activated.");
    SerialOnline = true;
  }
  else
  {
    println("\nI found "+PortCount+" serial ports, which are:");
    println(PortNumber);
    println(Serial.list());
    println("But I don't know which one to use. :(\n");
    println("Now entering offline simulation mode.\n");

  }



  println("Press keys '0' through '3' for angle XYZ, X, Y or Z.\n");
  background(255);
  noStroke();
  newPoint = false;
  DrawAxes = true;

  BoundariesInitialized = false;
 FGcolorVal = 0;

}

int CameraType = 0;
```

```
void draw() {

  if(keyPressed) {
   if(key >= '0' && key <= '3') {
     DrawAxes = true;

     if(key == '0') {
       CameraType = 0;
       }
            if(key == '1') {
       CameraType = 1;
       }
            if(key == '2') {
       CameraType = 2;
       }
            if(key == '3') {
       CameraType = 3;
       }
   }
  }


  lights();
  ambientLight(5,5,5);

  if (CameraType == 0){
    camera(300,-300,300,
    0.0, 0.0, 0.0,
    0, 0, -1);
  }

  if (CameraType == 1){
    camera(500,0,0,
    0.0, 0.0, 0.0,
    0, 0, -1);
  }

  if (CameraType == 3){
    camera(0,0,500,
```

```
  0.0, 0.0, 0.0,
  0, 1, 0);
}

if (CameraType == 2){
  camera(0,-500,000,
  0.0, 0.0, 0.0,
  0, 0, -1);
}

if (DrawAxes == true)
{
  DrawAxes = false;
background(255);
  fill(255, 0, 0);
  i = 0;
  while (i < 50){
    pushMatrix();
    translate(6*i,0,0);
    sphere(3);
    popMatrix();
    i++;
  }

  pushMatrix();
  translate(100,0,0);
  sphere(5);
  popMatrix();
  pushMatrix();
  translate(200,0,0);
  sphere(5);
  popMatrix();


  fill(0, 255, 0);

  i = 0;
  while (i < 50){
    pushMatrix();
```

```
    translate(0,-6*i,0);
    sphere(3);
    popMatrix();
    i++;
  }

  pushMatrix();
  translate(0,-100,0);
  sphere(5);
  popMatrix();
  pushMatrix();
  translate(0,-200,0);
  sphere(5);
  popMatrix();
  fill(0, 0, 255);
  i = 0;
  while (i < 50){
    pushMatrix();
    translate(0,0,6*i);
    sphere(3);
    popMatrix();
    i++;
  }

  pushMatrix();
  translate(0,0,100);
  sphere(5);
  popMatrix();
  pushMatrix();
  translate(0,0,200);
  sphere(5);
  popMatrix();
}
 if ( newPoint== true)
 {
 //  fill (250);

fill(FgColorFunc(1));
```

```
    newPoint = false;
    pushMatrix();

    translate(xpos/2, -1 * ypos/2, zpos/2);

    sphere(3);
    popMatrix();
  }
  else if ( SerialOnline == false)
  {
      fill (250);

    pushMatrix();
    rotateX(random(2*PI));
    rotateY(random(2*PI));
    rotateX(random(2*PI));
    translate(200,0,0);
    sphere(3);
    popMatrix();
  }

}

void mousePressed() {
        println(xpos + "\t" + ypos + "\t" + zpos + "\t" + tpos
          + "\t" + xMin  + "\t" + xMax
          + "\t" + yMin  + "\t" + yMax
          + "\t" + zMin + "\t" + zMax
          );
 println("Bx\tBy\tBz\t|B|\tBxMin\tBxMax\tByMin\tByMax\tBzMin\tBzMax");

float xt;
float yt;
float zt;
     xt = float((xMax + xMin)/2) + xOffs;
     yt =  float((yMax + yMin)/2) + yOffs;
     zt =  float((zMax + zMin)/2)  + zOffs;

         println("Correction factors:\nfloat xOffs =  " +   xt +
```

```
          ";\nfloat yOffs = " +   yt +
          ";\nfloat zOffs = " +  zt + ";\n");

  exit();
}


void serialEvent(Serial myPort) {  int inByte = myPort.read();
  if (firstContact == false) {
    if (inByte == 'A') {
      myPort.clear();
      firstContact = true;
myPort.write('A');
    }
  }
  else {
    serialInArray[serialCount] = inByte;
    serialCount++;

    if (serialCount > 5 ) {

      xpos = (serialInArray[0] << 8) | serialInArray[1];
      ypos = (serialInArray[2] << 8) | serialInArray[3];
      zpos = (serialInArray[4] << 8) | serialInArray[5];

      if (xpos > 32767)
      {
        xpos -= 32767;
        xpos = -xpos;
      }
      if (ypos > 32767)
      {
        ypos -= 32767;
        ypos = -ypos;
      }
      if (zpos > 32767)
      {
        zpos -= 32767;
        zpos = -zpos;
```

```
}

  newPoint = true;
  newX = (float)xpos/2048;
  newY =  (float)ypos/2048;
  newZ =  (float)zpos/2048;
  newX = (1000 * newX)  - xOffs;
  newY = (1000 * newY)  - yOffs;
  newZ = (1000 * newZ)  - zOffs;
  newT = sqrt(newX * newX + newY * newY + newZ * newZ);

  xpos = (int) round(newX);
  ypos = (int) round(newY);
  zpos = (int) round(newZ);
  tpos = (int) round(newT);

  if (BoundariesInitialized == false){
    BoundariesInitialized = true;

    xMin = xpos;
    xMax = xpos;
    yMax= ypos;
    yMin= ypos;
    zMax= zpos;
    zMin= zpos;
  }

  if (xpos < xMin)
    xMin = xpos;
  if (ypos < yMin)
    yMin = ypos;
  if (zpos < zMin)
    zMin = zpos;
  if (xpos > xMax)
    xMax = xpos;
  if (ypos > yMax)
    yMax = ypos;
  if (zpos > zMax)
    zMax = zpos;
```

```
println(xpos + "\t" + ypos + "\t" + zpos + "\t" + tpos
    + "\t" + xMin  + "\t" + xMax
    + "\t" + yMin  + "\t" + yMax
    + "\t" + zMin + "\t" + zMax
    );
  myPort.write('A');
  serialCount = 0;
 }
}
}
```

Conclusioni

Terminata la fase di progettazione e programmazione abbiamo testato personalmente il dispositivo all'interno dell'istituto ottenendo buoni risultati : quando il sonar rileva qualsiasi forma di ostacolo, segnala il pericolo al non vedente grazie ad un bip prodotto dal sonar che accelera all'avvicinarsi dell'ostacolo tramite un auricolare.
Il risultato ottenuto si è dimostrato coerente con lo scopo per cui era stato progettato il dispositivo.

Risultati

Attraverso l'inserimento di un carattere veranno visualizzati i diversi risultati in output dei relativi componenti sul monitor seriale:

- premendo il carattere "a" visualizzo la distanza calcolata dal primo sensore;

-premendo il carattere "b" visualizzo la distanza calcolata dal secondo sensore;

-premendo il carattere "m" visualizzo le coordinate in x,y,z.