RELAZIONE SONAR

Sala Alvise classe 5BS a.s. 2021-22

PROGETTO

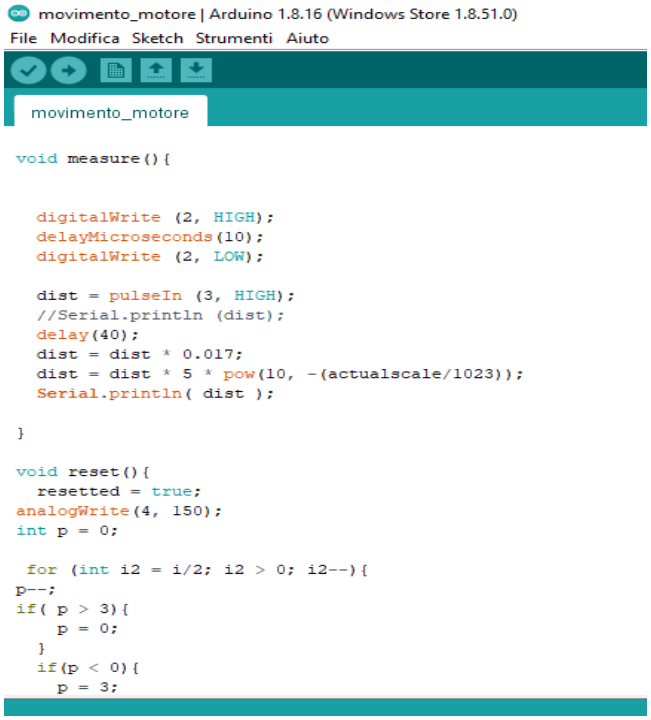
Creare un sonar in grado di disegnare su schermo una mappatura degli oggetti circostanti.

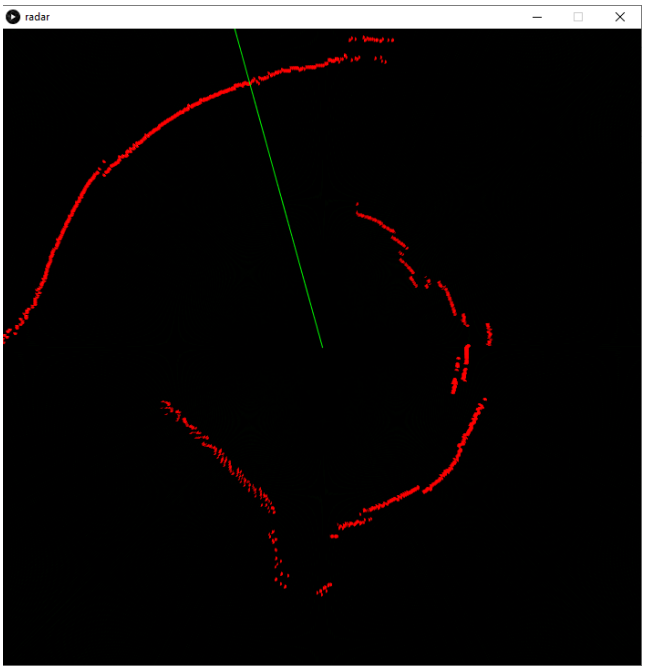
MATERIALI

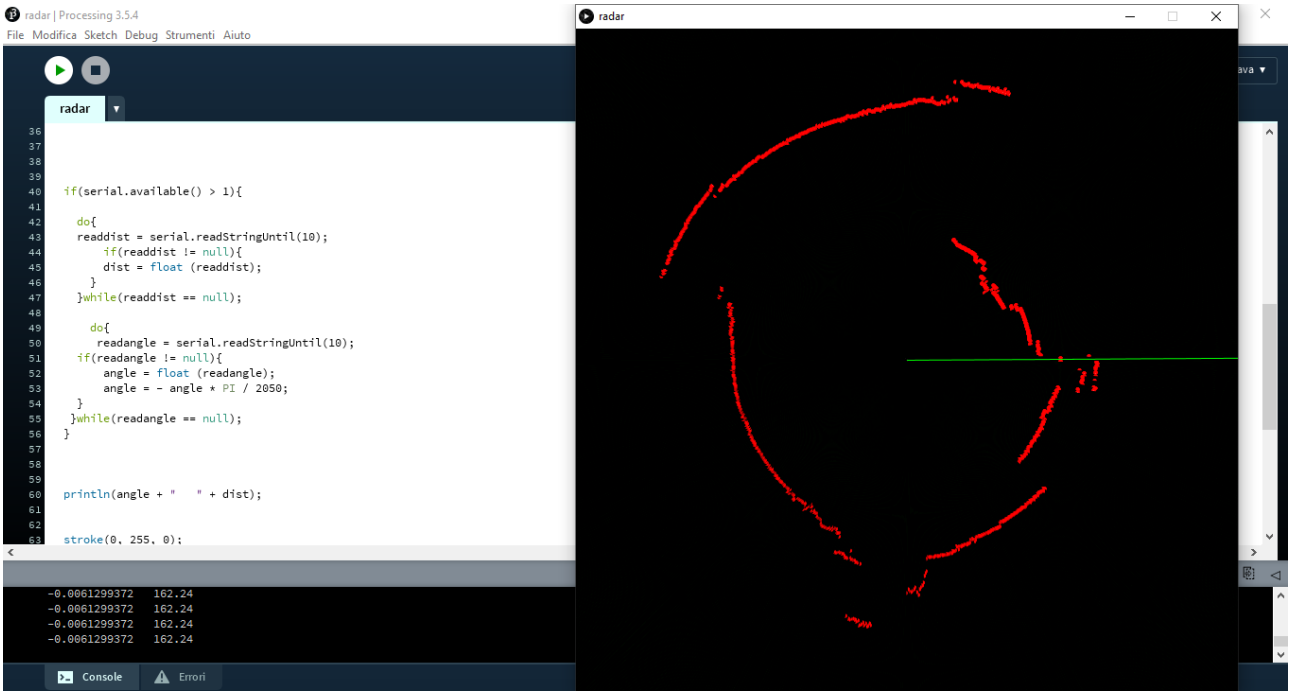
Scheda Arduino
Sensore di distanza SR04
Stepper motor
Scheda traduttrice motore
Mini breadboard
Potenziometro
Pulsanti, cavi, resistenze, led

DESCRIZIONE

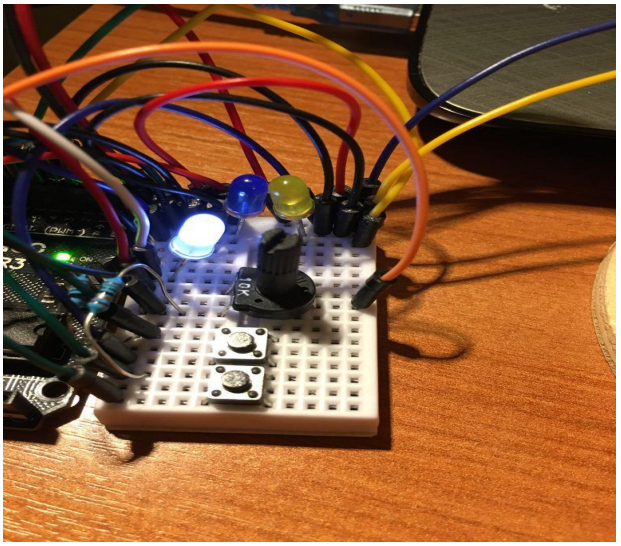
Il sonar è composto da un sensore ad ultrasuoni che, emettendo delle onde sonore di frequenza nota, permette di misurare la distanza da un ostacolo a cui sta puntando ed è montato su un motore che compie un giro di 360/4020 gradi ogni volta che viene effettuata una misurazione.
Il sensore SR04 ogni 45 s compie un giro completo di 360° acquisendo tutte le informazioni necessarie per disegnare a schermo tutti gli ostacoli che lo circondano in un raggio di 4 m.
L'intero meccanismo è controllato da 2 software che collaborano tra di loro: il programma Arduino gestisce il movimento del motore, i comandi esterni e le misurazioni e si occupa di mandare come dati in output sulla porta seriale l'angolo di rotazione del motore e la distanza misurata, mentre il software Processing acquisisce in input i dati e li usa per disegnare per ogni misurazione un punto con il corretto angolo di rotazione e distanza dal centro.







Il sonar è dotato di comandi esterni con cui manovrare il funzionamento del dispositivo: un pulsante aziona il giro, il secondo pulsante resetta la rotazione del motore e il potenziometro regola la scala con cui è visualizzata sul display la mappatura.



Tastiera dei comandi

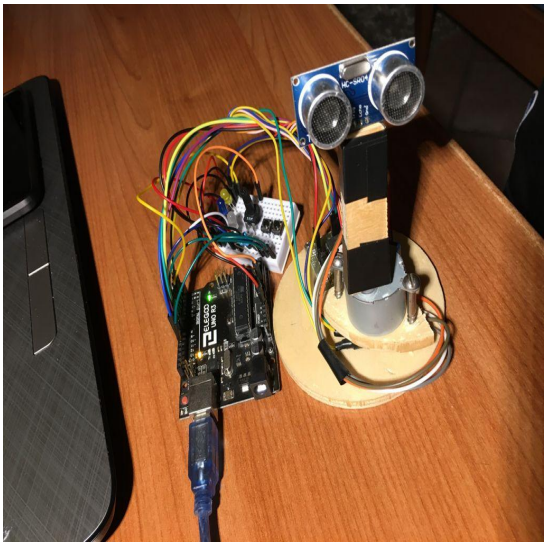


Foto del sonar

SOFTWARE ARDUINO

```
int x, t = 110, i = 1, t2 = 100;
bool spin = true;
bool sw = false, resetted = false;
float dist, angle, scale, actualscale = 1;


void setup() {
pinMode(12, INPUT);
pinMode(A0, INPUT);
pinMode(11, OUTPUT);
pinMode(10, OUTPUT);
pinMode(9, OUTPUT);
pinMode(8, OUTPUT);

pinMode(3, INPUT);
pinMode(2, OUTPUT);
```

```
digitalWrite(2, LOW);

Serial.begin(9600);
}

void loop(){

t++;
t2++;



 if((digitalRead(7) == 1) && (t >= 100)){
    sw = !sw;
    t = 0;
  }



  if(sw){
      spin = true;
      actualscale = scale;



      for (i=0; i<4100; i++){
        onespin();


        if((digitalRead(12) == 1) && (t2 >= 100)){
        reset();
        t2 = 0;
        break;
    }
```

```
  }


if (!resetted){
      delay (500);
      spin = false;



      for (i=4100; i>0; i--){
        onespin();

            if((digitalRead(12) == 1) && (t2 >= 100)){
            reset();
            t2 = 0;
            break;
        }
      }
    }
    else{
      resetted = false;
    }
  }
}

void onespin() {

 scale = analogRead(A0);

  if(scale > 30){
 analogWrite(5, scale/30);
```

```
}else{
  analogWrite(5, 1);
}

if(spin){
  x++;
}
else{
  x--;
}

if(x > 7){
  x = 0;
}
if(x < 0){
  x = 7;
}

switch (x){
  case 0:
  digitalWrite(8, HIGH);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, LOW);
  break;

  case 1:
  digitalWrite(8, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(10, LOW);
```

```
digitalWrite(11, LOW);
break;

case 2:
digitalWrite(8, LOW);
digitalWrite(9, HIGH);
digitalWrite(10, LOW);
digitalWrite(11, LOW);
break;

case 3:
digitalWrite(8, LOW);
digitalWrite(9, HIGH);
digitalWrite(10, HIGH);
digitalWrite(11, LOW);
break;

case 4:
digitalWrite(8, LOW);
digitalWrite(9, LOW);
digitalWrite(10, HIGH);
digitalWrite(11, LOW);
break;

case 5:
digitalWrite(8, LOW);
digitalWrite(9, LOW);
digitalWrite(10, HIGH);
digitalWrite(11, HIGH);
break;
```

```
  case 6:
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, HIGH);
  break;

  case 7:
  digitalWrite(8, HIGH);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, HIGH);
  break;
}



if(sw){
  analogWrite(6, 35);
}
else{
  digitalWrite(6, LOW);
}

t++;
t2++;

if((digitalRead(7) == 1) && (t >= 30)){
  sw = !sw;
  t = 0;
```

```
  }

if(i%4 == 0){
measure();
 Serial.println( i );
}

}

void measure(){


  digitalWrite (2, HIGH);
  delayMicroseconds(10);
  digitalWrite (2, LOW);

  dist = pulseIn (3, HIGH);
  //Serial.println (dist);
  delay(40);
  dist = dist * 0.017;
  dist = dist * 5 * pow(10, -(actualscale/1023));
  Serial.println( dist );

}

void reset(){
  resetted = true;
analogWrite(4, 150);
int p = 0;
```

```
 for (int i2 = i/2; i2 > 0; i2--){
p--;
if( p > 3){
    p = 0;
  }
  if(p < 0){
    p = 3;
  }

  switch (p){
    case 0:
    digitalWrite(8, HIGH);
    digitalWrite(9, LOW);
    digitalWrite(10, LOW);
    digitalWrite(11, LOW);
    break;

    case 1:
    digitalWrite(8, LOW);
    digitalWrite(9, HIGH);
    digitalWrite(10, LOW);
    digitalWrite(11, LOW);
    break;

    case 2:
    digitalWrite(8, LOW);
    digitalWrite(9, LOW);
    digitalWrite(10, HIGH);
    digitalWrite(11, LOW);
    break;
```

```
    case 3:
    digitalWrite(8, LOW);
    digitalWrite(9, LOW);
    digitalWrite(10, LOW);
    digitalWrite(11, HIGH);
    break;
  }
  delay(2);
 }

digitalWrite (4, LOW);

}
```

SOFTWARE PROCESSING

```
import processing.serial.*;

String readdist, readangle;
float dist, angle;
boolean clear = false;

Serial serial;
String port_name = Serial.list()[0];

void setup(){

  size (700, 700);
```

```
  background(0);



   println(Serial.list()[0]);


   serial = new Serial (this, port_name, 9600);
   frameRate(60);
}

void draw(){




    translate(350, 350);



  stroke(0, 0, 0);
  strokeWeight(2.5);
  circle (0, 0, 4);
  line (0, 0, 5000 * cos(angle), 5000 * sin(angle));

  stroke(255, 0, 0);
  circle(dist * cos(angle), dist * sin (angle), 2);




  if(serial.available() > 1){

    do{
```

```
  readdist = serial.readStringUntil(10);
      if(readdist != null){
      dist = float (readdist);
    }
  }while(readdist == null);

    do{
     readangle = serial.readStringUntil(10);
  if(readangle != null){
      angle = float (readangle);
      angle = - angle * PI / 2050;
  }
 }while(readangle == null);
}




println(angle + "   " + dist);



stroke(0, 255, 0);
strokeWeight(1);
line (0, 0, 5000 * cos(angle), 5000 * sin(angle));

  if(clear){
    fill(0);
    rect(-1000, -1000, 10000, 10000);
    clear = false;
  }
```

```
}


void keyPressed(){
  if (keyCode == 81){
  clear = true;
  }
}



void keyReleased(){
}
```