Michiel Vanschoonbeek (KUL)
Jonas Pelgrims (KUL)

Report: Sonar system

Physical Computing based on Open Software and Hardware Platforms

1. Project description

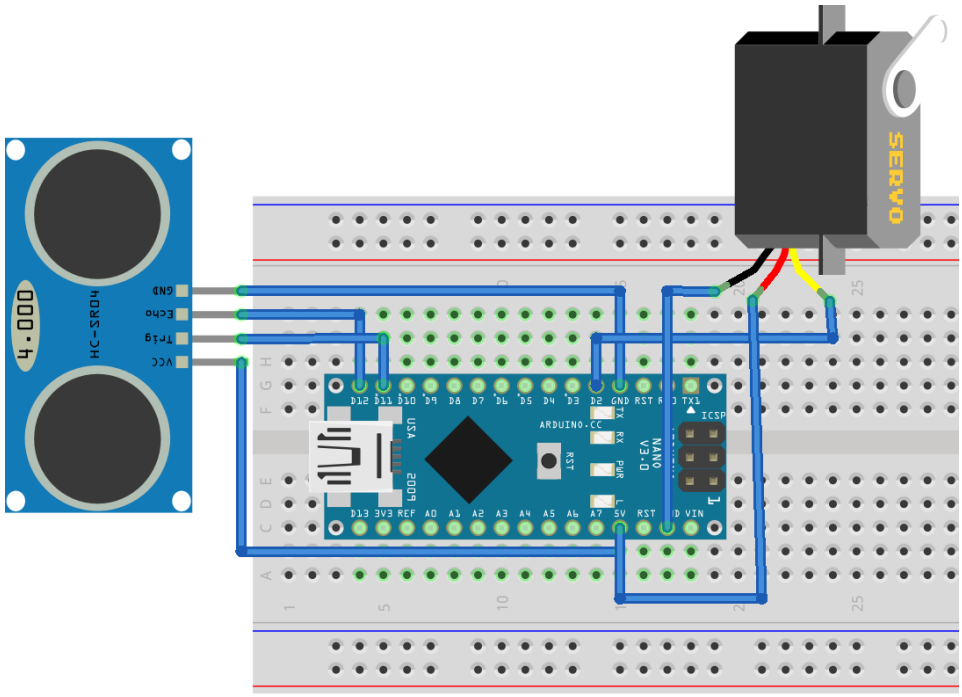
This report describes the design of a sonar system. For this project, we use a HC-SR04 ultrasonic ranging module and a SG90 Micro Servo. The ultrasonic sensor is mounted on top of the servo. In this way, it can turn 180 degrees. While turning, the environment is scanned and the data is transferred to a user application.

The application is written in Processing. The purpose is to visualize the data from the sensor on a radar system. The sensor turns 180 degrees and one measurement is done per angle. When turning back, the corresponding measurements are combined and the accuracy is improved. That feature is very useful for static environments. Without averaging the measurements, some data is inaccurate. If the first measurement is significantly different from the second, the data is discarded.

2. Materials used

- Breadboard
- Arduino Nano
- SG90 Micro Servo
- HC-SR04 Ultrasonic ranging module

3. Schematic



4. Code

4.1 Processing code

```
import processing.serial.*;


// declaration of variables used
PShape radar;
  int radius = 493;
  int centerX = 494;
  int centerY = 700;

  int MAX_DISTANCE = 100;
  int MIN_DISTANCE = 2;
  int MIN_RADAR_DISTANCE = 50;
  int MAX_RADAR_DISTANCE = 494;
  int POINT_THICKNESS = 8;

  int index_avg;
  int lf = 10;

  int angle;
  int distance;
  int brightness2;

  String comPortString;

  int[] angles = new int[180];
  int[] distances = new int[180];
  int[] angles2 = new int[180];
  int[] distances2 = new int[180];
  int[] distances3 = new int[180];




  Serial myPort; //defining Serial

  void setup() {
    size(1000, 700);

    radar = loadShape("radar.svg");
    //setting up the serial connection
    myPort = new Serial(this, Serial.list()[9], 9600);
    myPort.bufferUntil('\n');

  }

  void draw() {
    // drawing background and radar shape
    background(0);
    shape(radar, 0,200,1000,500);
    // drawing the text
    textSize(50);
    fill(0, 255, 0);
    text("Sonar Project", 10, 50);
    textSize(26);
    fill(0, 255, 0);
    text("Current angle: "+angles[0]+" degrees", 10, 130);
    fill(0, 255, 0);
    text("Current Distance: "+distances[0]+" cm ", 10, 160);
```

```
63.     fill(0, 255, 0);
64.     text(": First sweep", 700, 130);
65.     fill(0, 255, 0);
66.     text(": Average sweep ", 700, 160);
67.
68.     textSize(16);
69.     fill(0,255,0);
70.     text(MAX_DISTANCE+" CM", 470, 195);
71.     fill(0,255,0);
72.     text((MAX_DISTANCE/6)*4+" CM", 470, 330);
73.     fill(0,255,0);
74.     text((MAX_DISTANCE/6)*2+" CM", 470, 468);
75.
76.     //legende dots
77.     strokeWeight(6);
78.     stroke(brightness2);
79.     point(685, 152);
80.
81.     strokeWeight(POINT_THICKNESS);
82.     stroke(255);
83.     point(685, 122);
84.
85.     // drawing radar line
86.     drawLine(angle);
87.
88.
89.     // drawing all points who are relevant
90.     for (int i=0; i<angles.length; i++) {
91.       int brightness;
92.
93.       brightness = 255;
94.       brightness2 = 155;
95.       if (distances[i] != MAX_DISTANCE) {
96.         drawPoint(angles[i],distances[i], brightness, POINT_THICKNESS
   );
97.         drawPoint(angles[i],distances3[i],brightness2, 5);
98.
99.       }
100.      //drawPoint(angles[i],distances[i], brightness,
   POINT_THICKNESS);
101.      //drawPoint(angles[i],distances3[i],brightness2, 5);
102.    }
103.
104. }
105.
106.
107. // code to draw the radar lines by angle
108. void drawLine(int angle) {
109.
110.   float x,y;
111.
112.   float radian = radians(angle);
113.   x = radius * cos(radian);
114.   y = radius * sin(radian);
115.
116.   float px = centerX - x;
117.   float py = centerY - y;
118.
119.   stroke(255);
120.   strokeWeight( 5 );
121.   line(centerX, centerY, px, py);
```

```
122.
123.  }
124.
125.  // drawing the points by angle and distance and mapping it on the r
      adar
126.  void drawPoint(int angle, int distance, int brightness , int
      POINT_THICKNESS) {
127.
128.    distance = constrain(distance, MIN_DISTANCE, MAX_DISTANCE);
129.    distance = round(map(distance,MIN_DISTANCE,MAX_DISTANCE,MIN_RADAR
      _DISTANCE,MAX_RADAR_DISTANCE));
130.
131.    strokeWeight(POINT_THICKNESS);
132.    stroke(brightness);
133.    point(centerX-distance*cos(radians(angle)), centerY-
      distance*sin(radians(angle)));
134.  }
135.
136.  // here we recieve values given by the arduino and putting creating
       a radar effect so that the dots disapear when the radar line passes
137.  void serialEvent(Serial cPort) {
138.
139.    comPortString = cPort.readStringUntil('\n');
140.    if (comPortString != null) {
141.
142.      comPortString=trim(comPortString);
143.      String[] values = split(comPortString, ',');
144.
145.      try {
146.
147.        shiftArrays(); // created a shift array to keep the points
      displayed and make it disapear
148.        shiftArrays3();
149.
150.        angles[0] = Integer.parseInt(values[0]);
151.        angle = angles[0];
152.        distances[0] = Integer.parseInt(values[1]);
153.        distance = distances[0];
154.
155.        if (angle == 180 || angle == 0) {
156.          reverseArrays(); // when 180 degree the shift aray needs to
      by reversed to make the right ones dissapear
157.          angles2 = angles.clone(); // making copies for the average
      calculations
158.          distances2 = distances.clone();
159.        }
160.
161.      index_avg = pos2val(angles2,angle);
162.
163.      if (distances2[index_avg] != MAX_DISTANCE) {
164.        distances3[0] = (distances2[index_avg] + distance)/2; //
      calculating avg distance when relevant
165.
166.      }
167.      else {
168.        distances3[0] =  distance;
169.      }
170.
171.
172.      } catch (Exception e) {}
173.    }
```

```
174.
175. }
176.
177. void shiftArrays() {
178.   for (int i=angles.length-1; i>=1; i--) {
179.
180.     distances[i] = distances[i-1];
181.     angles[i] = angles[i-1];
182.
183.   }
184. }
185. void shiftArrays3() {
186.   for (int i=angles.length-1; i>=1; i--) {
187.
188.     distances3[i] = distances3[i-1];
189.
190.   }
191. }
192.
193.
194. void reverseArrays()
195. {
196.     int len = angles.length;
197.     int len2 = len >> 1;
198.     int temp;
199.     int temp2;
200.     int temp3;
201.
202.     for (int i = 0; i < len2; ++i)
203.     {
204.         temp = angles[i];
205.         angles[i] = angles[angles.length - i - 1];
206.         angles[angles.length - i - 1] = temp;
207.
208.         temp2 = distances[i];
209.         distances[i] = distances[angles.length - i - 1];
210.         distances[angles.length - i - 1] = temp2;
211.
212.         temp3 = distances3[i];
213.         distances3[i] = distances3[angles.length - i - 1];
214.         distances3[angles.length - i - 1] = temp3;
215.
216.     }
217. }
218.
219. int pos2val(int[] angle2, int angle) {
220.   for (int i = 0; i <angle2.length ; i++) {
221.     if (angle2[i] == angle) return i;
222.
223.   }
224.  return 0 ;
225.
226. }
```

4.2 Arduino code

```
#include <Servo.h>
Servo myservo;
int PIN_12 = 12;
int PIN_11 = 11;
int angle = 1;
int angle_step = 1;
int upperbound = 180;
int lowerbound = 0;
int distance;
boolean lfr = 1;
#include <NewPing.h>

#define TRIGGER_PIN  11
#define ECHO_PIN     12
#define MAX_DISTANCE 100

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

long duration, inches, cm;

void setup() {

  Serial.begin(9600);
  myservo.attach(3);
}

void loop() {
  delay(50);

  if (angle >= upperbound or angle <= lowerbound) {
    lfr =!lfr;
  }
  if (lfr == 1) {
    angle += angle_step;
  }
  else {
    angle -= angle_step;
  }
  distance = sonar.ping_cm();
  if (distance == 0) {
    distance = MAX_DISTANCE;
  }
  Serial.print(angle);
  Serial.print(",");
  Serial.println(distance);
  myservo.write(angle);

}
```

5. Photographs and screenshots

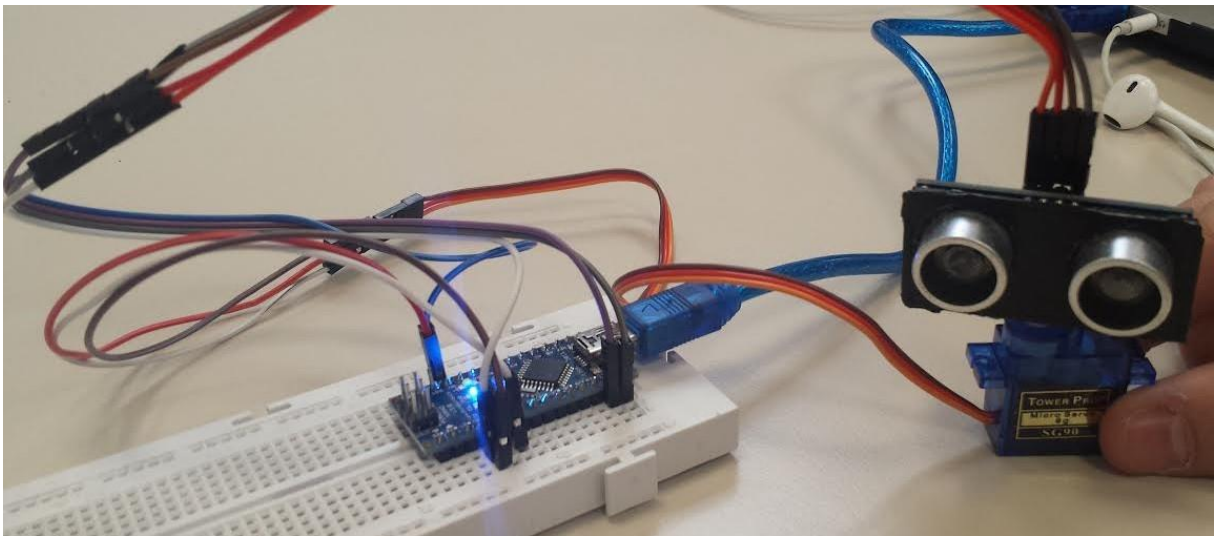


Figure 1 Hardware and materials used

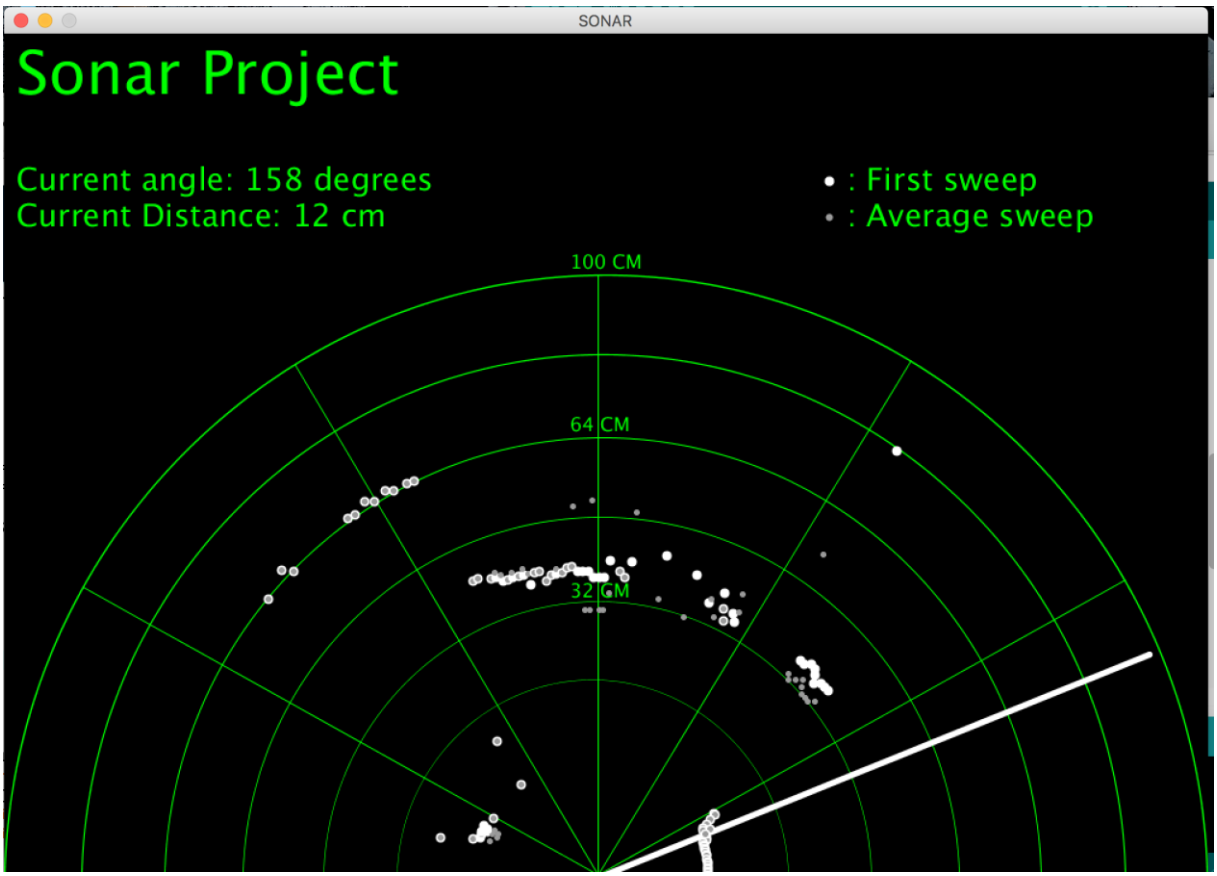


Figure 2: The application

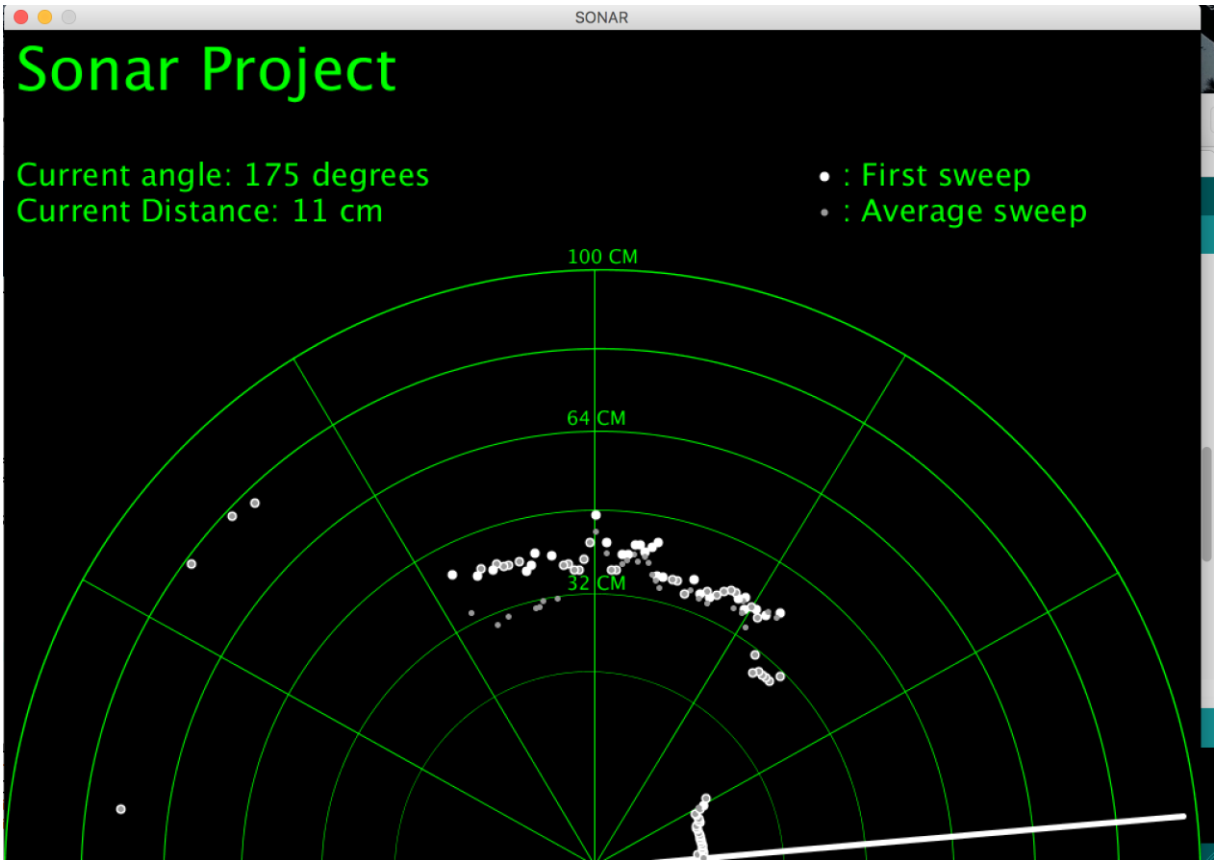


Figure 3: The application in operation