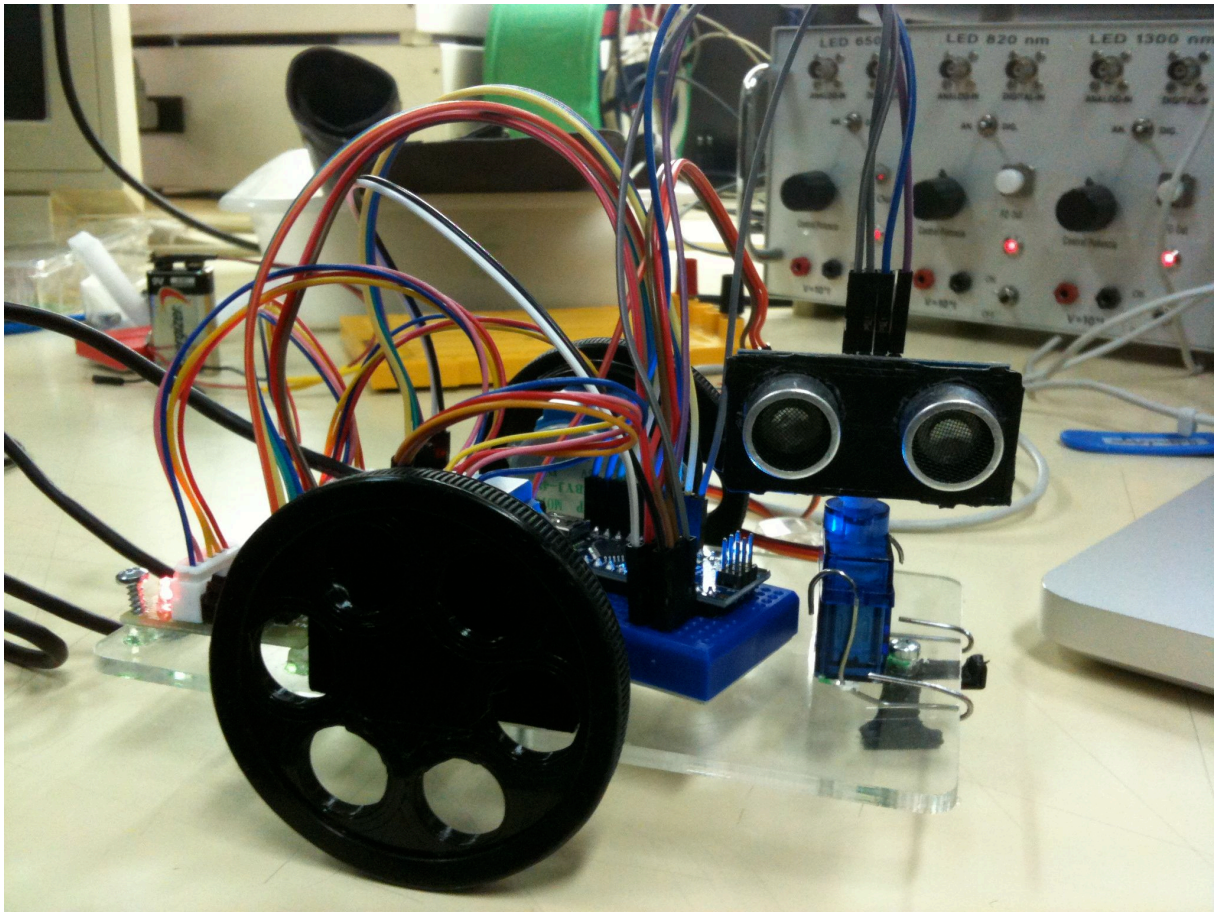
ARDUINO LAB PROJECT

ATHENS Programme, Universidad Politécnica de Madrid

MINES ParisTech

November 14-18, 2016

Nils Holzenberger Mathieu Prouveur



Project Description

Our project is a robot that can drive around and scan its environment. The robot can be controlled to go forward and backward and to spin around itself on its 2 wheels. The scan feature uses the sonar to measure the distance to the environment in a semicircle in front of the robot.

The project makes use of an Arduino Nano chip, stepper and servo motors, a sonar, some 3D printed pieces, and other mechanical components.

Hardware

All the components were mounted onto a 10x15 cm Plexiglas rectangle. We use an Arduino Nano ATmega328. Two 28BYJ-48 stepper motors, controlled with the OliStepper library, are used to spin the wheels. The sonar is controlled with the NewPing library, and is glued to an SG90 servomotor. The servo is controlled with the Servo library.

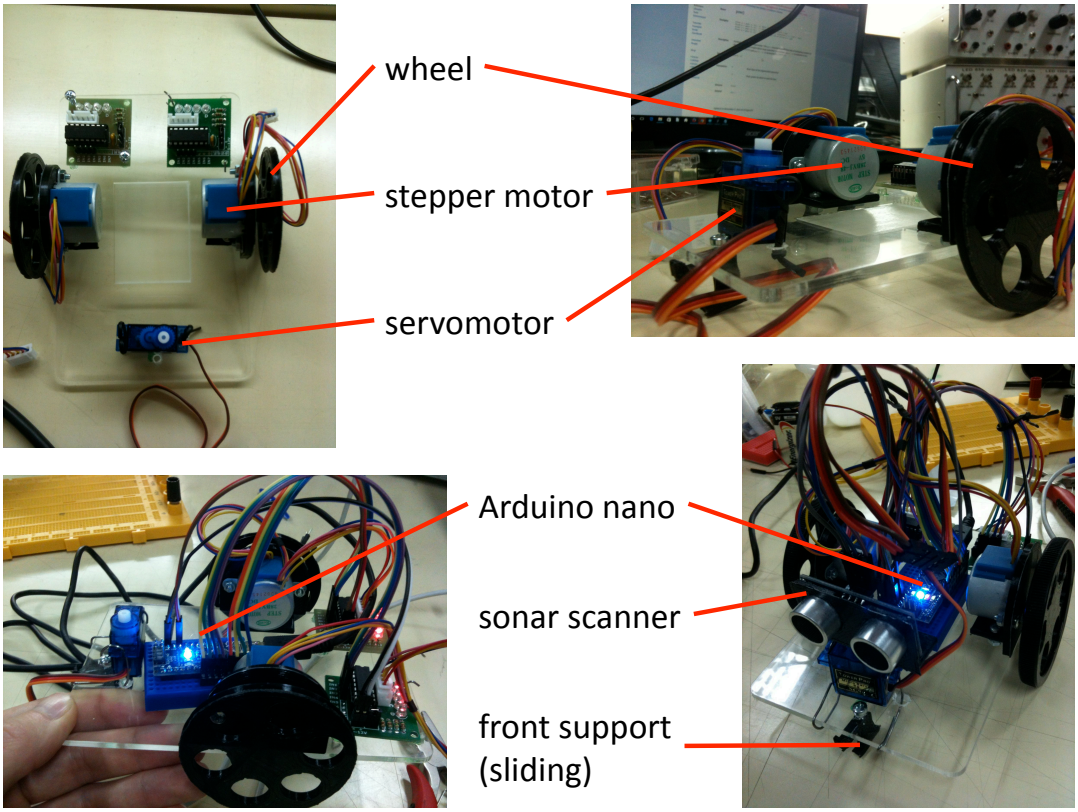
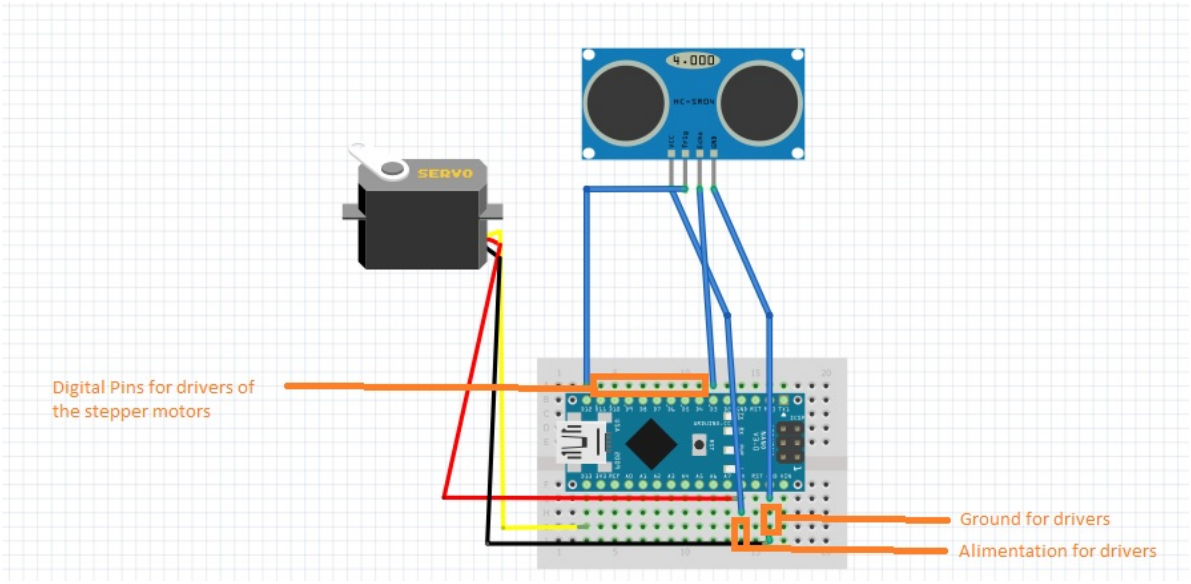


Figure 1. Components of the robot.



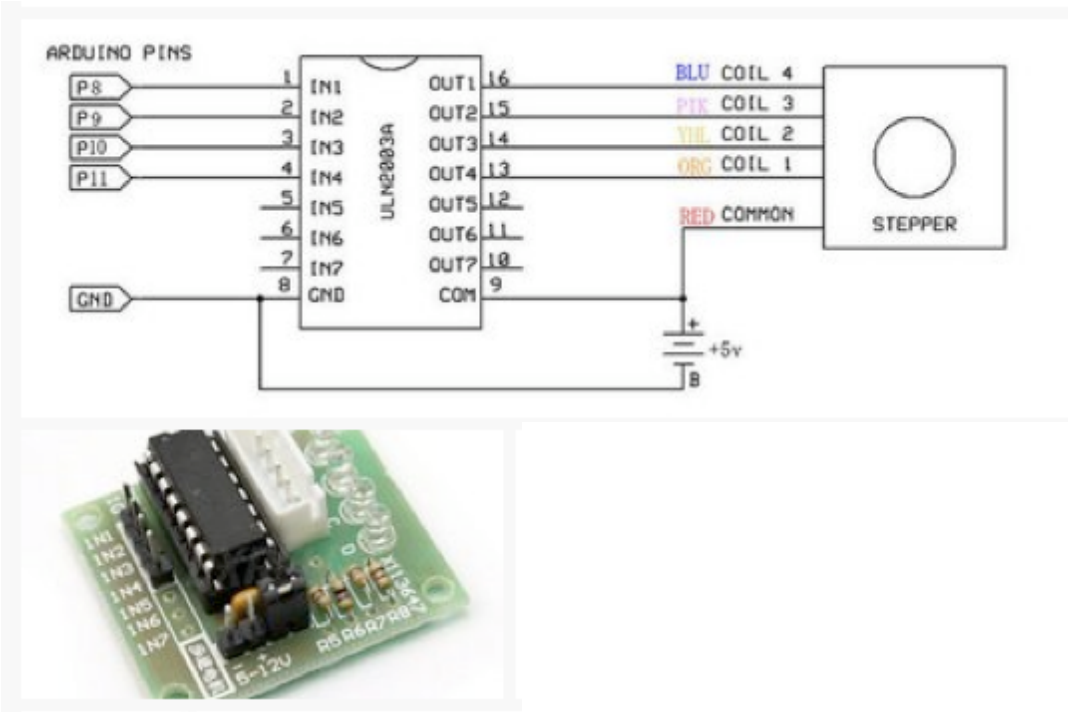


Figure 2. Schematic of the wirering.

GUI

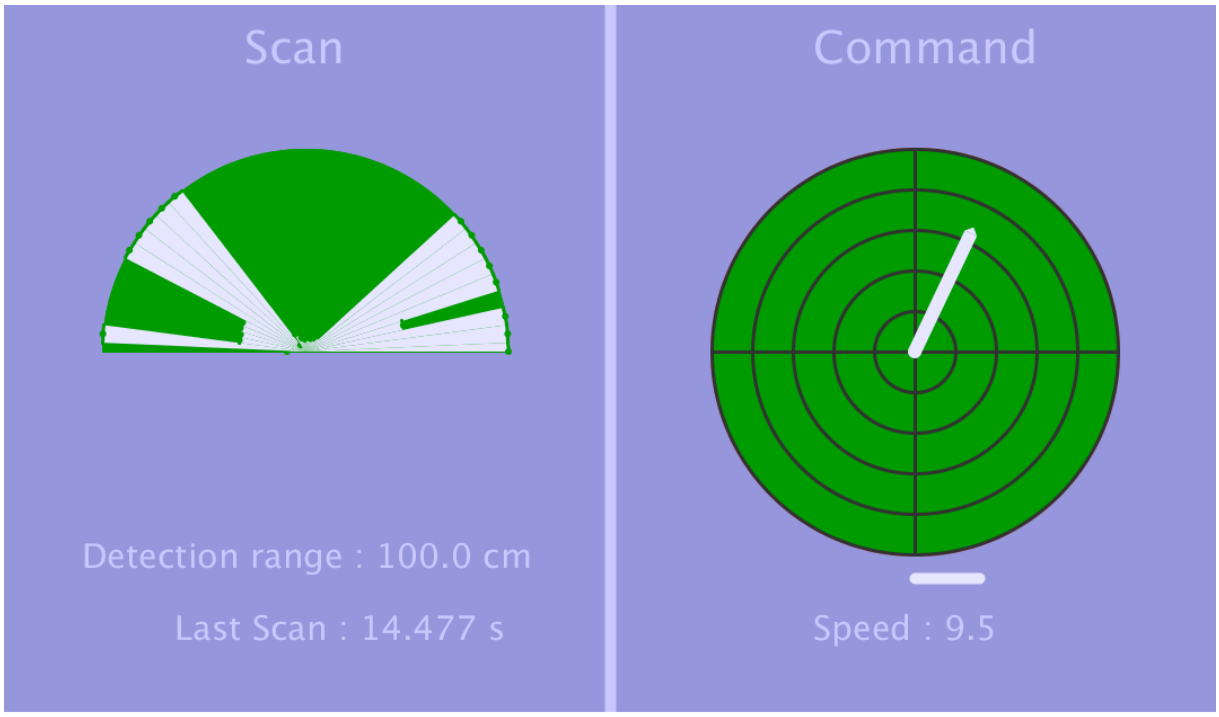


Figure 3. Graphical User Interface. Tab to scan, Enter to stop the robot, arrow keys to control speed and direction.

The GUI was made in Processing. The semicircle on the left indicates the last scan that was performed. The white sectors show the distance for a given angle. If the sonar does not detect anything within its range, we pretend there is an object at the end of the range.

The circle on the right shows the command that is being given to the robot. The arrow indicates magnitude and direction of the speed. The bar at the bottom shows whether the robot is turning or not (angular speed).

The speed of the robot can be adjusted by pressing on the keyboard arrows. Pressing once on a key increases the robot's speed in that direction. Pressing enter stops the robot, pressing tab prompts it to perform a scan. During scanning, the robot stops, and resumes its course afterwards.

Arduino and Processing code

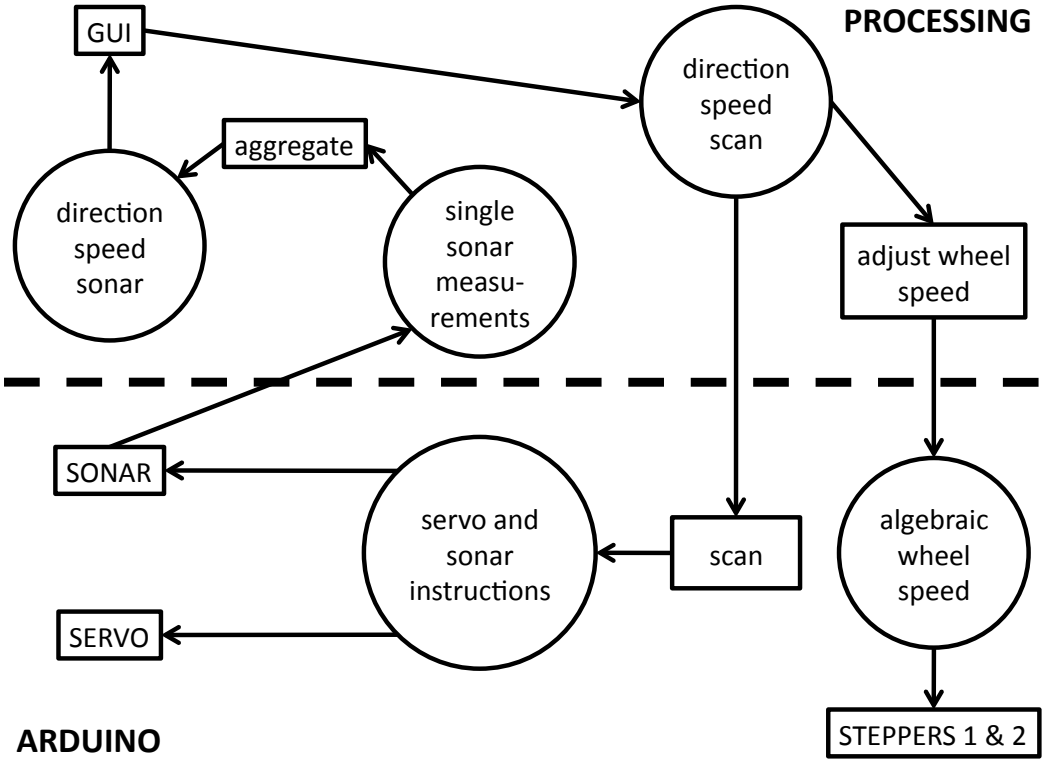


Figure 4. Structure of the code. Circles are values, lower case boxes are functions, upper case boxes are objects. The dashed line divides the code that is on the Arduino chip from the code executed from the computer.

The software follows a master/slave approach. Instructions from the GUI are translated into motor commands by Processing and passed to the Arduino. Data is read from the Arduino and formatted in Processing.

Conclusion

The robot can go forward and backward and spin on itself. We would have liked to implement a way for the robot to take a turn like a car, e.g. drive in a circle. Simply spinning one wheel faster than the other in the same direction caused the robot to stop.

Coordinating the sonar and the servo works perfectly. A bit of extra processing was necessary when receiving the data from the Arduino, because the data is sent in chunks.

We are very happy with the project. It was quite challenging to plug all those elements together, and also a lot of fun to make it work. With more time, we would have tried to implement wifi communication with the robot.

Code

1. Arduino

```
#include <OliStepper.h>
#include <Servo.h>
#include <NewPing.h>

float ANGLE_STEP = 5;
float MAX_DISTANCE = 100; // in cm
OliStepper stepper_r(8, 9, 10, 11, 512);
OliStepper stepper_l(4, 5, 6, 7, 512);
float rpm_l;
float rpm_r;
Servo servo;
NewPing sonar(12, 3, MAX_DISTANCE);
//boolean scanning = false;
String instructions = "";
String inst_buffer = "";

void set_rpm(float l, float r) {
  rpm_l = l;
  rpm_r = r;
}

void update_steppers() {
  if (rpm_r > 0) {
    stepper_r.setDirection("CW");
  }
  else {
    stepper_r.setDirection("CCW");
  }
  stepper_r.setRPM(abs(rpm_r));
  if (rpm_l < 0) {
    stepper_l.setDirection("CW");
  }
  else {
    stepper_l.setDirection("CCW");
  }
  stepper_l.setRPM(abs(rpm_l));
}

void oli_run() {
  stepper_l.oliRun();
```

```
  stepper_r.oliRun();
}

void update_instructions(String serial_message) {
  inst_buffer += serial_message;
  int first_C = inst_buffer.indexOf('C');
  int last_C = inst_buffer.lastIndexOf('C');
  while (first_C < last_C) {
    inst_buffer = inst_buffer.substring(first_C + 1);
    first_C = inst_buffer.indexOf('C');
    last_C = inst_buffer.lastIndexOf('C');
  }
  int first_A = inst_buffer.indexOf('A');
  instructions = inst_buffer.substring(first_A, first_C + 1);
}

void move_and_send(int pos) {
  servo.write(pos);
//  float distance = sonar.convert_cm(sonar.ping_median(4));
    float distance = sonar.convert_cm(sonar.ping());

  String m=String(distance)+"S";
  Serial.print(m);
  // TODO send the distance
  //analogWrite(3, distance);
}

void scan() {
  servo.write(180);
  Serial.write('S');
  delay(50);
  for (int pos = 180; pos >= 0; pos -= ANGLE_STEP) {
    delay(50);
    move_and_send(pos);
  }
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  servo.attach(13);
  servo.write(90);
  rpm_l = 0.0; // no movement to begin with
  rpm_r = 0.0;
  update_steppers();
}

void loop() {
  // put your main code here, to run repeatedly:
  //servo.write(30);
```

```
  oli_run();
  if (Serial.available() > 0) {
    String message = Serial.readString();
    if (message[0]=='S') {
      set_rpm(0.0, 0.0);
      update_steppers();
      scan();
      servo.write(90);
    }
    else {
      update_instructions(message);
      Serial.println(message);
      int ind = instructions.indexOf("B");
      String inst_l = instructions.substring(1, ind);
      String inst_r = instructions.substring(ind + 1, instructions.length() - 1);
      float new_rl = inst_l.toFloat();
      Serial.print(new_rl);
      float new_rr = inst_r.toFloat();
      Serial.print(new_rr);
      if (new_rl != rpm_l | new_rr != rpm_r) {
        set_rpm(new_rl, new_rr);
        update_steppers();
      }
    }
  }
}
```

2. Processing

```
float vg, vd,length;
PFont fontA;

void setup() {
  size(1080, 640);
  vg=0.0;
  vd=0.0;
  println("vg= " + vg + "  vd= "+ vd);


  background(150, 150, 220);
  // Set the font and its size (in units of pixels)
}

void draw() {
  background(150, 150, 220);
  stroke(255, 100, 100);
    strokeWeight(3);
```

```
  fill(150, 150, 220);
  ellipse(width/2, height/2, width/3, width/3);
  stroke(50);

  for( int i=0;i<5;i++){
  ellipse(width/2, height/2, width/3-i*width/15, width/3-i*width/15);
  }

  stroke(50);
  line(width/2-width/6, height/2,width/2+width/6, height/2);
  line(width/2, height/2-width/6,width/2, height/2+width/6);
  drawV(vg,vd);
}

void keyPressed() {

  if (key==CODED) {
    if (keyCode==LEFT) {
      vd+=1.0;
      vg-=1.0;
    }
    if (keyCode==UP) {
      vd+=1.0;
      vg+=1.0;
    }
    if (keyCode==DOWN) {
      vd-=1.0;
      vg-=1.0;
    }
    if (keyCode==RIGHT) {
      vd-=1.0;
      vg+=1.0;
    }
    vg=min(14,vg);
    vg=max(-14,vg);
    vd=min(14,vd);
    vd=max(-14,vd);
    println("vg= " + vg + "  vd= "+ vd);
  }
  if (keyCode== ENTER) {
    vg=0.0;
    vd=0.0;

    println("vg= " + vg + "  vd= "+ vd);
  }

}
```

```
void drawV(float vg, float vd){

  pushMatrix();
  fill(155,90,100);
  length=width/6*(abs(vg+vd)/28.0);
  noStroke();
  translate(width/2,height/2);

  if(vg+vd!=0){
    if(vd<=0&&vg<=0){
  rotate(PI);
  rotate(-atan((vg-vd)/(vg+vd)));
  }
  else{
    rotate(atan((vg-vd)/(vg+vd)));
  }}


  triangle(-width/180.0,-length+width/180,width/180,-length+width/180,0,-length);
  rect(-width/180,0,width/90,-length+width/180);
  ellipse(0,0,width/90,width/90);
  translate(width/2,height/2);
  popMatrix();

}
```