République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ECOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT D'ÉLECTRONIQUE

Mémoire de Master

Thème :

Application de la télémétrie ultrasonore pour la réalisation d'un radar

Encadré par : **M. MEHENNI**

Réalisé par : **DERAREDJ RAMZI**

Soutenu le 13 juin 2015 devant le jury

B. BOUSSEKSOU	Maitre Assistant A	ENP	Président
H. BOUSBIA-SALAH	Maitre de Conférences A	ENP	Examinateur
M. MEHENNI	Professeur	ENP	Promoteur

Promotion : Juin 2015

DEDICACE

Je dédie ce travail à :

Ma mère, qui a fait de moi l'homme que je suis

Mon père, mon modèle dans la vie

Mon grand frère Rafik qui m'a toujours soutenu et protégé

Mon adorable petite sœur Racha

Mon seul amour Meriem Rachi Inès

A mes meilleurs amis Rym Lahlou et Yacine abada et Sami ben abdalakhouja

Tous les membres de ma famille

Tous mes amis sans exception.

.DERAREDJ RAMZI.

REMERCIEMENTS

Je tiens donc en premier lieu à remercier mon encadreur *Mr M. MEHENNI, Professeur à l'ENP,* pour le temps et la patience qu'il m'a accordés tout au long de ce semestre. Il est un professeur très dévoué qui a répondu à mes nombreuses questions. Je garderai dans mon cœur sa pédagogie, son efficacité et sa rigueur. Pour tout ce qu'il m'a donné, qu'il trouve ici toute ma gratitude.

Je remercie *Mr H. BOUSBIA-Salah* et *Mr B.BOUSSEKSOU* qui ont accepté d'évaluer mon travail.

Je n'oublie pas non plus tous mes camarades de l'ENP en particulièrement mon ami Lenouar Alizinelabidine, un très grand merci à tous.

Enfin, je remercie mes parents, mes grands-parents, ma sœur, mon frère et toute ma famille de m'avoir encouragé et soutenu moralement tout au long de mes études, pour leur amour, leur confiance et leur présence dans les moments difficiles.

ملخص

فهو يصف طلب للحصول على الروبوت بالموجات فوق الصوتية الباحث .ويندرج هذا المشروع ضمن تطبيقات اردوينو

.عن نطاق شنت على المحرك لخلق خريطة للبيئة كشاشة نصف

ويتم الارتباط إلى دقة عرض عالية على جهاز الكمبيوتر مع تجهيز بيئة ممتازة

كلمات مفتاحية: تطبيقات اردوينو, فوق الصوتية, جهاز الكمبيوتر,الروبوت بالموجات

ABSTRACT

This project is part of Arduino's applications. It describes an application for a robot with ultrasonic rangefinder mounted on a servo to create a map of environment like an half display of true radar. The link to high resolution display on the PC is made with the Processing environment.

Keywords: Arduino, ultrasonic rangefinder, true radar, display on the PC

RESUME

Ce projet s'inscrit dans le cadre des applications Arduino. Il décrit une application pour un robot avec télémètre à ultrasons monté sur un servomoteur pour créer une carte de l'environnement comme un demi-écran. Le lien à l'affichage haute résolution sur le PC se fait avec l'environnement Processing.

Mots clés : Arduino , télémètre à ultrasons, application pour un robot, affichage sur le PC.

Liste des figures

TABLE DES MATIERES

1-INTRODUCTION

Ce projet s'inscrit dans le cadre du sujet master qui s'intéressera à l'extension d'une partie de notre projet de fin d'études qui traite de la télémétrie à ultrasons.

Ce document décrit un système de vision pour un robot avec télémètre à ultrasons monté sur un servomoteur pour créer une carte de l'environnement avec une présentation du résultat sur écran d'ordinateur simulant un vrai demi écran radar (vision vers l'avant à 180 degrés) via une application.

2-PRINCIPE DE FONCTIONNEMENT

Ce projet met en œuvre deux éléments principaux : un capteur ultrason à faible coût le HC-RS04 et un servomoteur à faible coût le RS-03.

Le fonctionnement de ce radar va être autonome, les résultats vont être transférés de la carte arduino vers un ordinateur via une liaison série par un câble USB, ces données vont être transmises à une application développée avec le logiciel Processing qui est un langage de programmation basé sur la plate-forme Java pour l'affichage graphique des résultats.

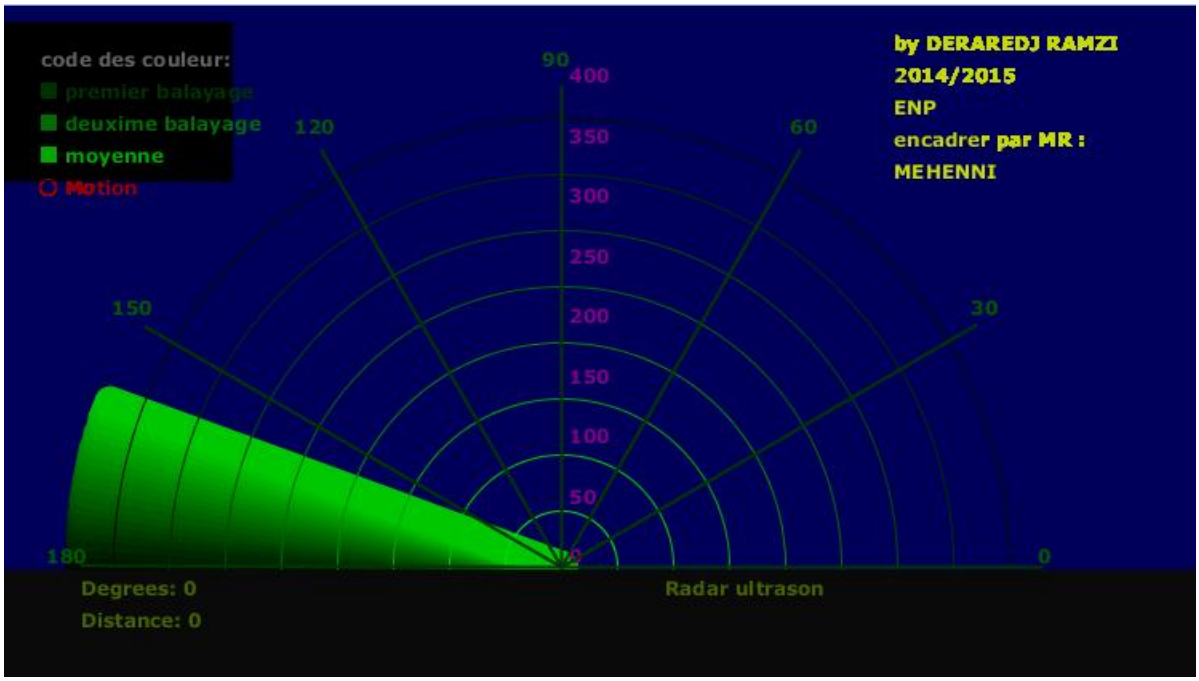


Figure 1 : Ecran obtenu du radar

La rotation du servomoteur permet un balayage permanent sur 180 degrés avec un pas de 10 degrés car le module à ultrason a un angle d'ouverture d'environ 15 degrés en favorisant la vision dans l'axe pendant les déplacements.

3- REALISATION

Dans la réalisation de ce projet il y a deux parties distinctes, le logiciel embarqué sur l'Arduino qui comporte la prise en main du servomoteur et les algorithmes de balayage privilégiant la vision axiale. La prise en main du capteur ultrasons et les algorithmes pour la détermination des distances ainsi que les échanges bidirectionnels entre Arduino et l'application sont développés par Processing à 9600bits.

Le logiciel en Processing sur PC est un langage et un environnement de programmation open source, c'est aussi le nom du compilateur. Il est utilisé par les étudiants, les artistes, les créateurs les chercheurs et les amateurs . Ce langage basique offre de très nombreuses possibilités et laisse libre cours à l'imagination, notamment grâce à une centaine de librairies disponibles. Le logiciel fonctionne sur Macintosh, sous Windows et sous Linux. En effet il est basé sur la plate-forme Java. Processing offre la possibilité de travailler en 3D, des jeux en réseau, des effets de son et lumière et bien d'autres applications. Par souci de cohérence, les parties communes ont la même terminologie de variables.

3-1. **PARTIE ARDUINO**

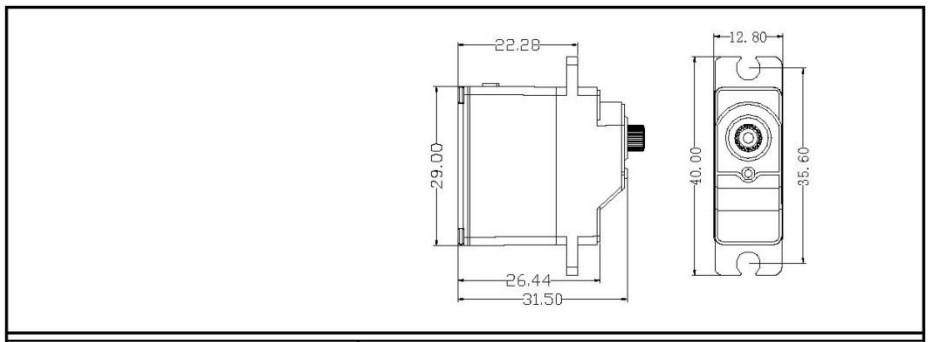
Le programme développé et injecté dans le microcontrôleur fait que le système fonctionne d'une manière autonome.

3-1-1 Le servomoteur

Tout d'abord, il faut savoir qu'un servomoteur permet d'affecter des déplacements en rotation. De nombreux objets du quotidien utilisent des servomoteurs: voitures télécommandées, drones, électroménager, etc. Les différents types de servomoteur se distinguent par trois caractéristiques:

- leur vitesse de rotation par seconde,
- leur course en degré,
- leur couple exprimé en kg.cm,

Pour notre projet on a utilisé le servomoteur RS-3 dont la fiche technique est représentée sur la figure 1.



Operating Voltage:		4.8~6.0V
Stall Torque:	4.8V	2.2 kg.cm(30.8 oz/in.)
	6.0V	2.5 kg.cm(35.0 oz/in.)
Operating Speed:	4.8V	0.11 sec/60 at no load
	6.0V	0.09 sec/60 at no load
Grade:		Hobby
Weight:		16.2g (0.56oz.)

Figure 2 : Spécifications du SERVO RS-3

3-1-1-1.Principe de fonctionnement du servomoteur

Le fonctionnement d'un servomoteur est relativement simple. Il suffit d'envoyer une impulsion dont la durée est comprise entre 0,5ms et 2,5ms au servomoteur de manière périodique, généralement toutes les 50ms. Cette impulsion déterminera la position du guide du servomoteur. Ainsi la valeur de 1,5ms donne au servomoteur la position centrale.

Le branchement d'un servomoteur est simple. Il a généralement trois fils:

- un rouge: qui doit être branché au 5V (ou 12V) suivant le servomoteur.
- un noir : qui doit être branché à la masse.
- un orange (ou autre couleur) : qui est le fil de la tension de commande 0-5V.

C'est donc, via le fil orange que les commandes seront passées à notre servomoteur. Ce fil devra obligatoirement être relié à votre Arduino sur l'entrée PWN.

Voici le montage qu'on a réalisé pour tester notre mini servomoteur:

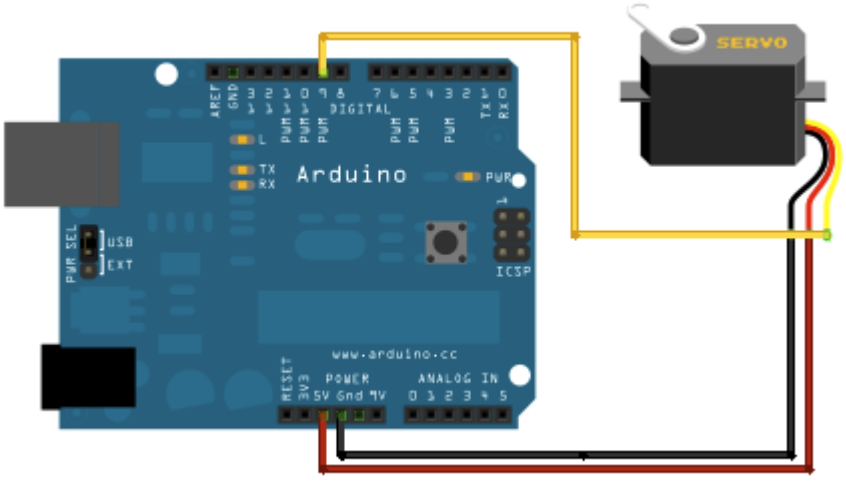


Figure 3 : connexion entre servomoteur et arduino

3-1-1-2. Programmation du servomoteur

Pour écrire l'algorithme de balayage qui donne une vision de 180 degrés, on va utiliser la librairie « Servo.h » et créer une variable de type « servo » qui va être attachée à la pin 9 de l'arduino. Pour faire tourner le servomoteur, on va utiliser l'instruction nom.Write (angle-valeur).

La portion de code ci-dessous se charge de faire tourner le servomoteur de 0 degré à180 degrés puis la deuxième boucle le fait tourner de 180° à 0 degré avec un pas qui va dépendre de l'angle d'ouverture de notre capteur ultrasons qui est pour notre cas 15 degrés.

```
#include <Servo.h>
Servo leftRightServo;
intleftRightPos=0;
void setup(){
leftRightServo.attach(9);
Serial.begin(9600);
}
void loop(){
for(leftRightPos=0;leftRightPos<180;leftRightPos)
{
```

```
leftRightServo.write(leftRightPos);
leftRightPos=leftRightPos+4;
}
for(leftRightPos=180;leftRightPos>0;leftRightPos)
{
leftRightServo.write(leftRightPos);
leftRightPos=leftRightPos-4;
}
}
```

3-1-2 Module à ultrasons

Comme il est bien expliqué dans le mémoire de mon projet de fin d'études, on ne va pas trop détailler la partie présentation du module et on s'intéressera au code qui va permettre de commander ce module et le faire interagir avec l'application développée par le logiciel Processing.

3-1-2-1.Principe de fonctionnement

Le module HC-SR04 est constitué de 4 Pins qui sont définies comme ci-dessous :

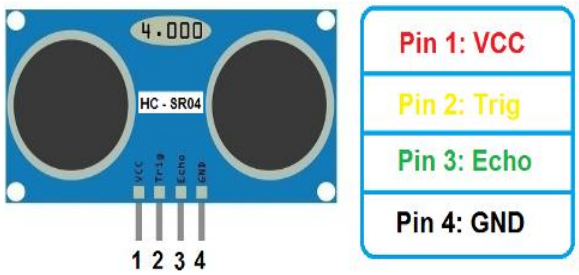


Figure 4 :Le module HC-SR04

•VCC : alimentation en 5V

•Trig : permet d'envoyer une impulsion de durée minimum 10µs pour demander une mesure.

•Echo : Une fois qu'une mesure est demandée, un signal carré dont la durée équivaut au temps mis par l'onde pour un aller et retour apparait sur cette pin.

•GND : la masse

Données techniques:

•Plage de mesure : de 2 cm à 400 cm

•Consommation max : 15mA

•Angle : <15°

Précision théorique : 3mm

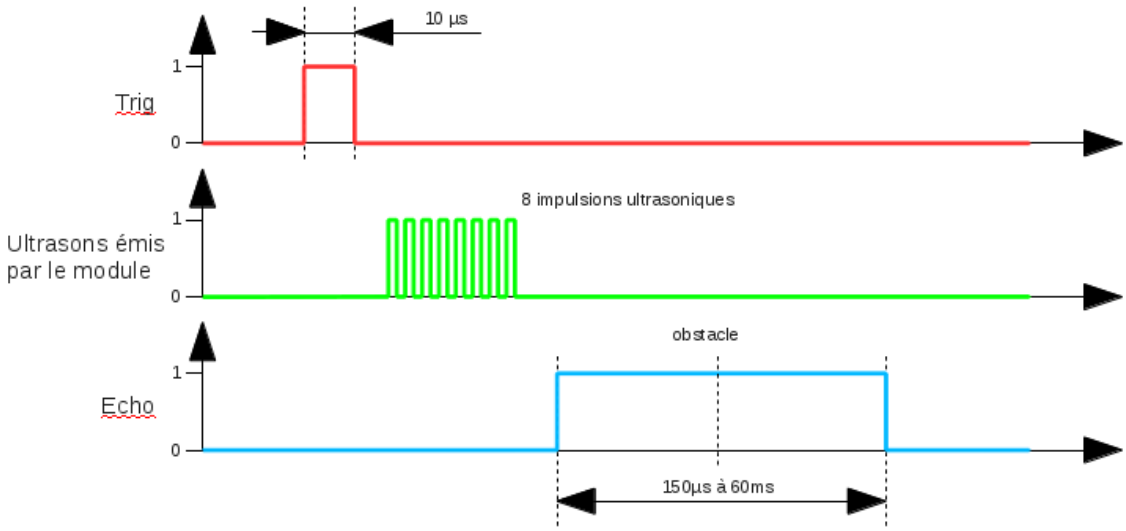


Figure 5 : chronogrammes illustrant le fonctionnement du module.

Quand le microcontrôleur fait passer la broche TRIG à l'état haut pour une durée de 10µs, le module émet une onde sonore composée d'une série de 8 impulsions à 40 kHz qui est inaudible pour l'oreille humaine, puis on active la broche Echo sur front montant et ensuite on calcule le temps que l'onde fait pour un aller-retour.

3-1-2-2.Programmation

On va s'intéresser à la partie qui va nous permettre de communiquer les résultats à l'application et pour cela il faut fixer la vitesse de l'échange bidirectionnelle entre l'arduino et l'application à 9600 bits/s dans la fonction d'initialisation. La communication est réalisée par voie série donc, il faut que l'arduino renvoie les informations sur la distance et l'angle vers l'ordinateur.

```
#include <Servo.h>

Servo leftRightServo;
intleftRightPos=0;
constintnumReadings=10;

intechoPin=10;
intinitPin=11;
```

```
unsignedlongpulseTime=0;
unsignedlong distance =0;

void setup(){
leftRightServo.attach(5);
pinMode(initPin, OUTPUT);
pinMode(echoPin, INPUT);
Serial.begin(9600);
}
```

L'idée consiste à renvoyer un message via la liaison série avec la lettre 'V' avant la valeur de l'angle et la lettre 'X' avant la valeur de la distance. L'application intercepte le message et le traite. On détaillera tout cela dans la partie processing.

```
void loop(){
for(leftRightPos=0;leftRightPos<180;leftRightPos){
leftRightServo.write(leftRightPos);
   delay(200);
digitalWrite(initPin, HIGH);
delayMicroseconds(10);
digitalWrite(initPin, LOW);
pulseTime=pulseIn(echoPin, HIGH);
    distance =pulseTime/58+3;

if(distance>400){distance=400;}
Serial.print("X");
Serial.print(leftRightPos);
Serial.print("V");
Serial.println(distance);
leftRightPos=leftRightPos+5;
}
for(leftRightPos=180;leftRightPos>0;leftRightPos){
```

```
leftRightServo.write(leftRightPos);
   delay(200);
digitalWrite(initPin, LOW);
digitalWrite(initPin, HIGH);
delayMicroseconds(10);
digitalWrite(initPin, LOW);
pulseTime=pulseIn(echoPin, HIGH);
    distance =pulseTime/58+3;

if(distance>400){distance=400;}
Serial.print("X");
Serial.print(leftRightPos);
Serial.print("V");
Serial.println(distance);
leftRightPos=leftRightPos-5;}}
```

3-2. PARTIE PROCESSING (AFFICHAGE SUR ORDINATEUR)

3-2-1 Présentation de l'interface logicielle

Dans cette partie, on va réaliser un environnement graphique approchant au mieux un écran radar conventionnel avec son balayage.

L'interface graphique se compose donc d'une barre d'outils, d'une entrée de texte, où on fait entrer le code, et une sortie de texte, où l'on pourra afficher du texte ainsi que les erreurs.

La barre d'outils est composée de 6 boutons :

Run : permet de lancer le programme pour son exécution.
Stop : permet d'arrêter l'exécution du programme, de fermer la fenêtre du programme.
New : permet de créer une nouvelle fenêtre vierge.
Open : permet d'ouvrir un programme existant, enregistré ou bien les exemples de création Processing préenregistrés.
Save : permet de sauvegarder votre fichier.

Le dernier bouton exporte l'application : pour créer sa propre application.

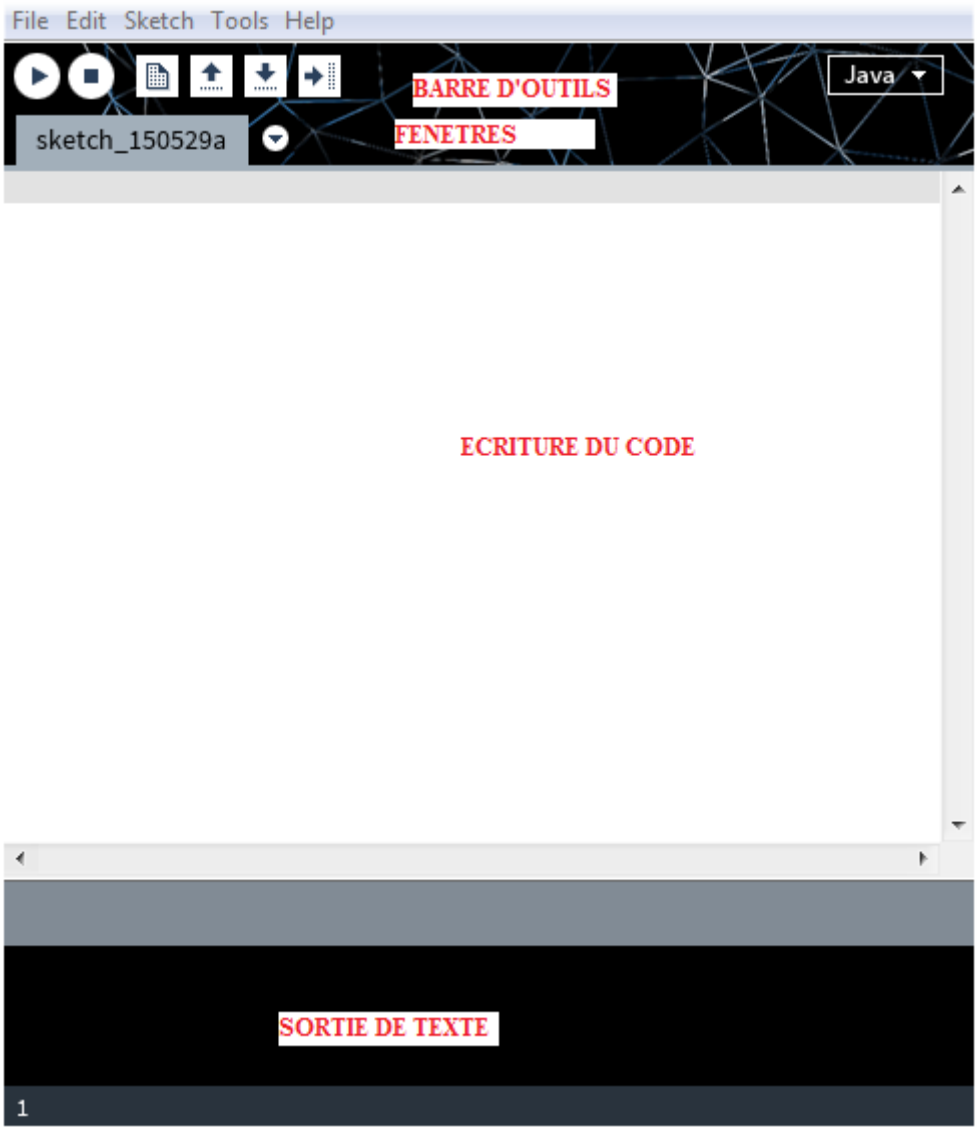


Figure 6 : L'interface graphique de Processing

3-2-2 Programmation

En premier lieu on va importer la bibliothèque « processing.serial.* » qui va nous permettre d'avoir une communication série avec l'arduino.

La programmation en processing est plus complexe que celle avec l'arduino.

3-2-2-1 Partie statique

On commence par la création d'une variable de type série que l'on va nommer 'arduinoport', sans oublier la déclaration des variables globales de notre code.

```
importprocessing.serial.*;
Serial arduinoport;
float x, y;
int radius =350;
int w =300;
int degree =0;
int value =0;
int motion =0;
int[]newValue=newint[181];
int[]oldValue=newint[181];
PFontmyFont;
intradarDist=0;
intfirstRun=0;
```

Après on passe à la fonction d'initialisation setup(), comme avec l'arduino, elle prend en charge l'initialisation du programme comme la taille de la fenêtre, la couleur et la vitesse de communication avec l'arduino qui est pour notre cas 9600 bits/seconde.

```
void setup(){
size(750,450);
background (0);
myFont=createFont("verdana",12);
textFont(myFont);
arduinoport=new Serial(this,Serial.list()[0],9600);
}
```

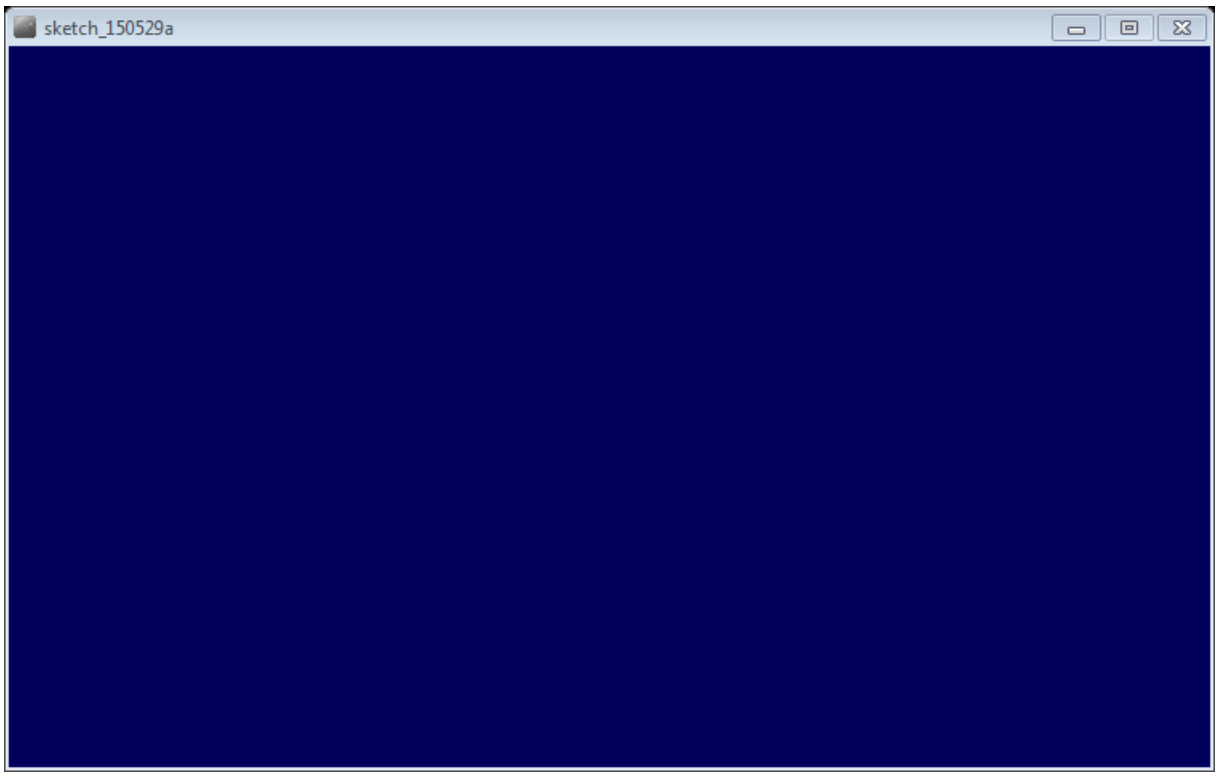


Figure 7: Résultat du code ci-dessus

La deuxième fonction « draw() » est la plus complexe. Elle est le cœur du programme car elle s'occupe du design.

On va expliquer chaque partie de notre code : on commence par créer un demi-cercle pour représenter notre radar avec une couleur bleu et une étiquète en bas dans laquelle il y a quelques informations.

```
void draw(){
fill(0,0,90);
noStroke();
ellipse(radius, radius,750,750);
rectMode(CENTER);
rect(350,402,800,100);
if(degree >=179){
motion =1;
}
if(degree <=1){
```

```
motion =0;
}}
noStroke();
fill(10);
rect(350,402,800,100);
fill(70,100,0);
text("Degrees: "+Integer.toString(degree),100,380,100,50);
text("Distance: "+Integer.toString(value*4/3),100,400,100,50);
text("Radar ultrason ",540,380,250,50);
fill(0);
```

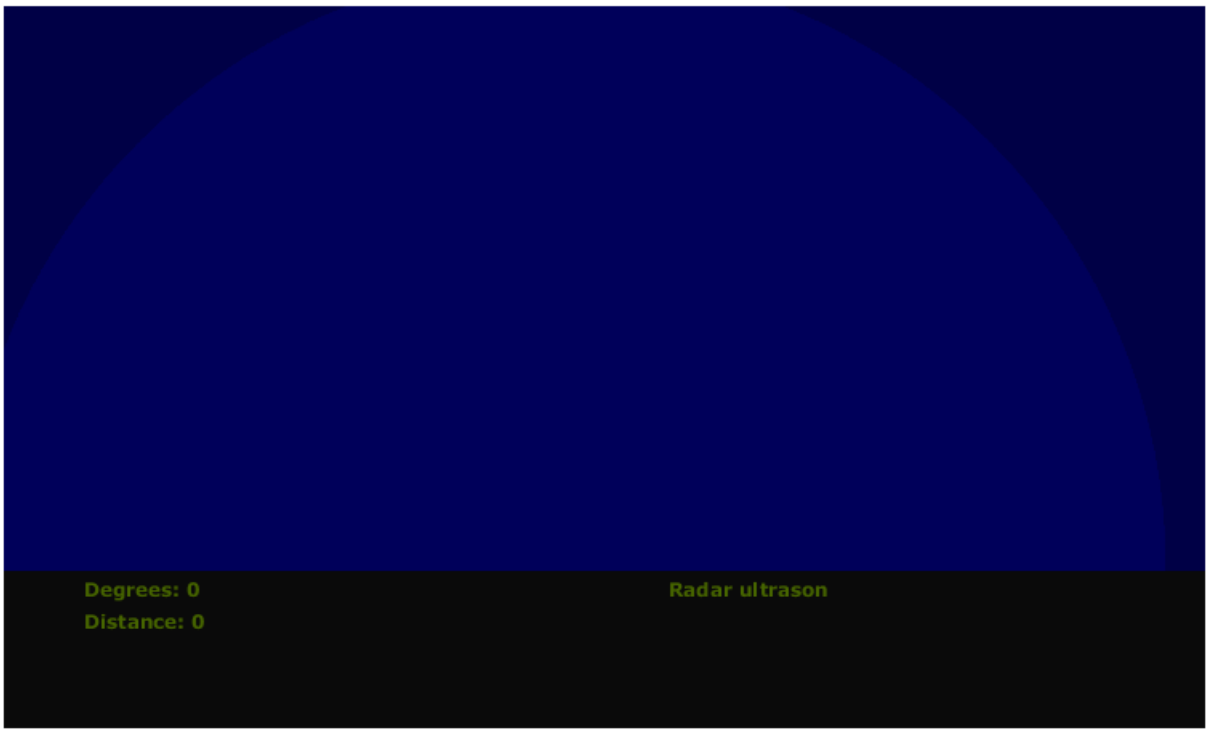


Figure 8: Résultat du code ci-dessus

Pour entrer les huit cercles représentant des distances constantes (étalonnage), on a le code suivant : c'est une boucle qui s'exécute huit fois.

```
for(int i =0; i <=8; i++){
noFill();
strokeWeight(1);
```

```
stroke(0,255-(30*i),0);
ellipse(radius, radius,(70*i),(70*i));
fill(130,0,130);
noStroke();
text(Integer.toString(radarDist),380,(360-radarDist*3/4),50,50);
radarDist+=50;
}
radarDist=0 ;
```

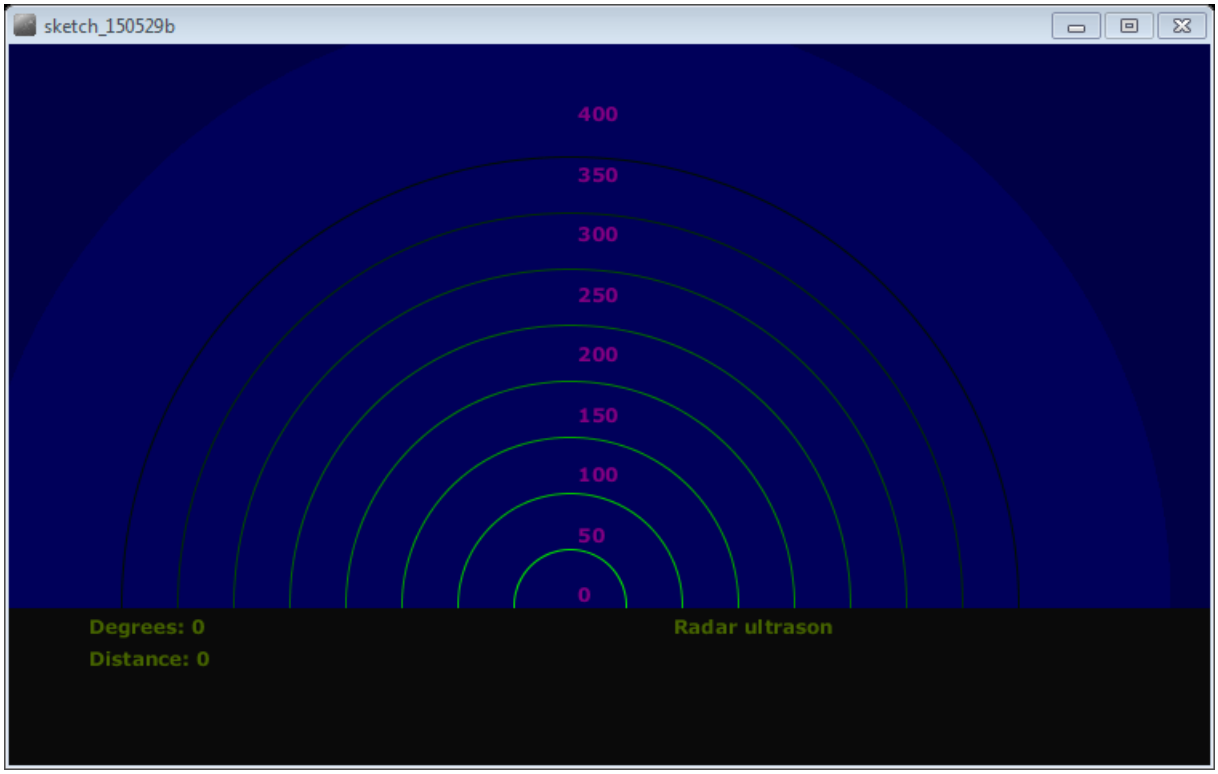


Figure 9: Résultat du code ci-dessus

Cette partie du code s'occupe des commentaires et des explications ajoutés sur l'écran.

```
for(int i =0; i <=6; i++){
strokeWeight(1);
stroke(0,55,0);
line(radius, radius, radius +cos(radians(180+(30*i)))*w, radius +
sin(radians(180+(30*i)))*w);
fill(0,55,0);
```

```
noStroke();
if(180+(30*i)>=300){
text(Integer.toString(180-
(30*i)),(radius+10)+cos(radians(180+(30*i)))*(w+10),(radius+10)+
sin(radians(180+(30*i)))*(w+10),45,50);
}else{
text(Integer.toString(180-(30*i)), radius +cos(radians(180+(30*i)))*w, radius +
sin(radians(180+(30*i)))*w,60,40);
}
}
```

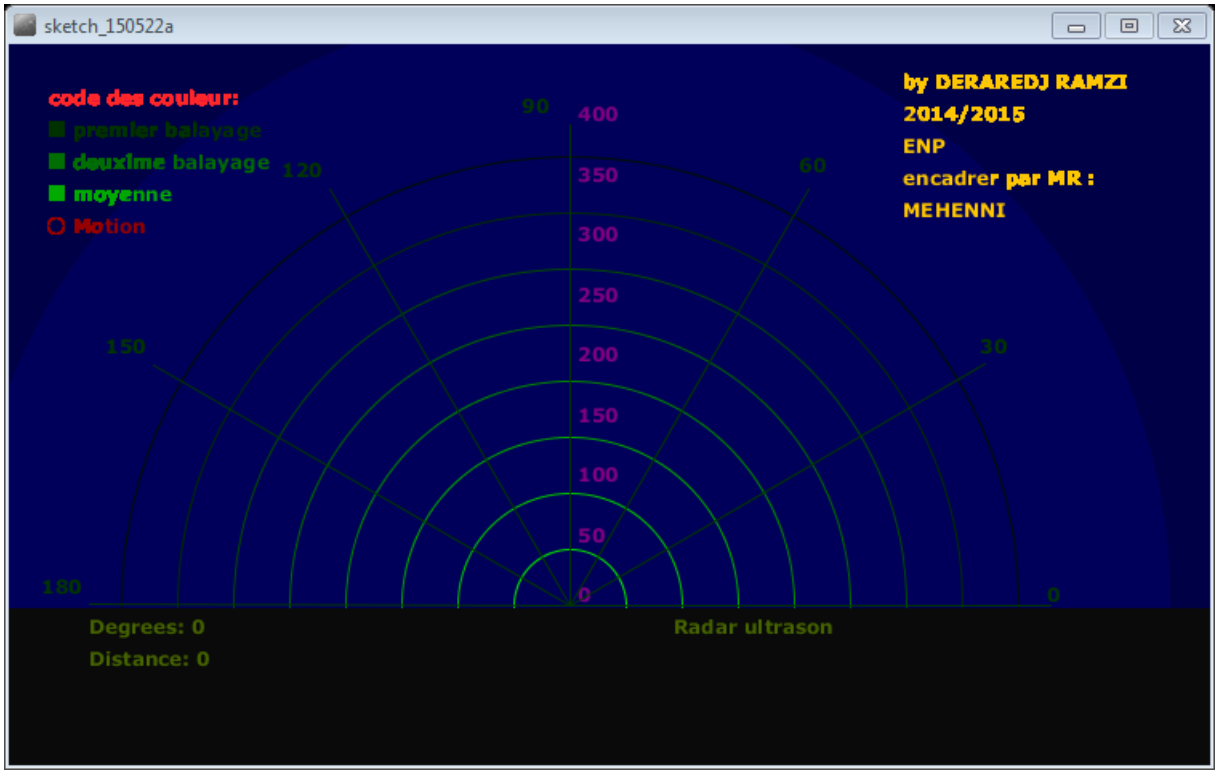


Figure 10: Résultat du code ci-dessus

3-2-2-2. Partie dynamique

Après avoir terminé la partie statique du code, on passe à la partie dynamique qui concerne la communication série de l'arduino avec l'application et le balayage de la zone de travail.

Pour la communication série on écrit une fonction qui intercepte les messages qui sont sous la forme ‘‘V70X180’’ où 70 est l’angle et 180 est la distance en centimètres. Cette fonction va chercher dans la voie série jusqu’à ce qu’elle trouve la lettre ‘V’, là elle va commencer à enregistrer les données et elle va s’arrêter une fois la lettre ‘’X’’ trouvée ; la donnée enregistrée et introduite dans une variable qui représente l’angle, le même travail pour la distance. Le code suivant illustre cette procédure.

```
voidserialEvent(Serial arduinoport){
String xString=arduinoport.readStringUntil('\n');
if(xString!=null){
xString= trim(xString);
String getX=xString.substring(1,xString.indexOf("V"));
String getV=xString.substring(xString.indexOf("V")+1,xString.length());
degree =Integer.parseInt(getX);
value =Integer.parseInt(getV);
oldValue[degree]=newValue[degree];
newValue[degree]= value;
firstRun++;
if(firstRun>360){
firstRun=360;
}}}
```

Cette partie permet l’écriture du code qui s’occupe du balayage, la première partie va concerner l’angle d’émission car il vaut 15 degrés ; donc pas besoin de faire un balayage avec un pas trop petit.

On a réalisé un programme nous permettant d’augmenter la probabilité de détection et de diminuer les erreurs en faisant trois balayages avec mémoire ce qui va permettre de trouver la cible avec précision.

```
trokeWeight(7);
if(motion ==0){
for(int i =0; i <=20; i++){
```

```
stroke(0,(10*i),0);
line(radius, radius, radius +cos(radians(degree+(180+i)))*w, radius +
sin(radians(degree+(180+i)))*w);}
}else{
for(int i =20; i >=0; i--){
stroke(0,200-(10*i),0);
line(radius, radius, radius +cos(radians(degree+(180+i)))*w, radius +
sin(radians(degree+(180+i)))*w);}}
fill(0,50,0);
beginShape();
for(int i =0; i <180; i++){
x = radius +cos(radians((80+i)))*((oldValue[i]));
y = radius + sin(radians((80+i)))*((oldValue[i]));
vertex(x, y);}
endShape();
fill(0,110,0);
beginShape();
for(int i =0; i <180; i++){
x = radius +cos(radians((180+i)))*(newValue[i]);
y = radius + sin(radians((180+i)))*(newValue[i]);
vertex(x, y);}
endShape();
fill(0,170,0);
beginShape();
for(int i =0; i <180; i++){
x = radius +cos(radians((180+i)))*((newValue[i]+oldValue[i])/2);
y = radius + sin(radians((180+i)))*((newValue[i]+oldValue[i])/2);
vertex(x, y);}
endShape();
if(firstRun>=360){
stroke(150,0,0);
strokeWeight(1);
noFill();
for(int i =0; i <180; i++){
```

```
if(oldValue[i]-newValue[i]>35||newValue[i]-oldValue[i]>35){
x = radius +cos(radians((180+i)))*(newValue[i]);
y = radius + sin(radians((180+i)))*(newValue[i]);
ellipse(x, y,10,10);
}}}
```

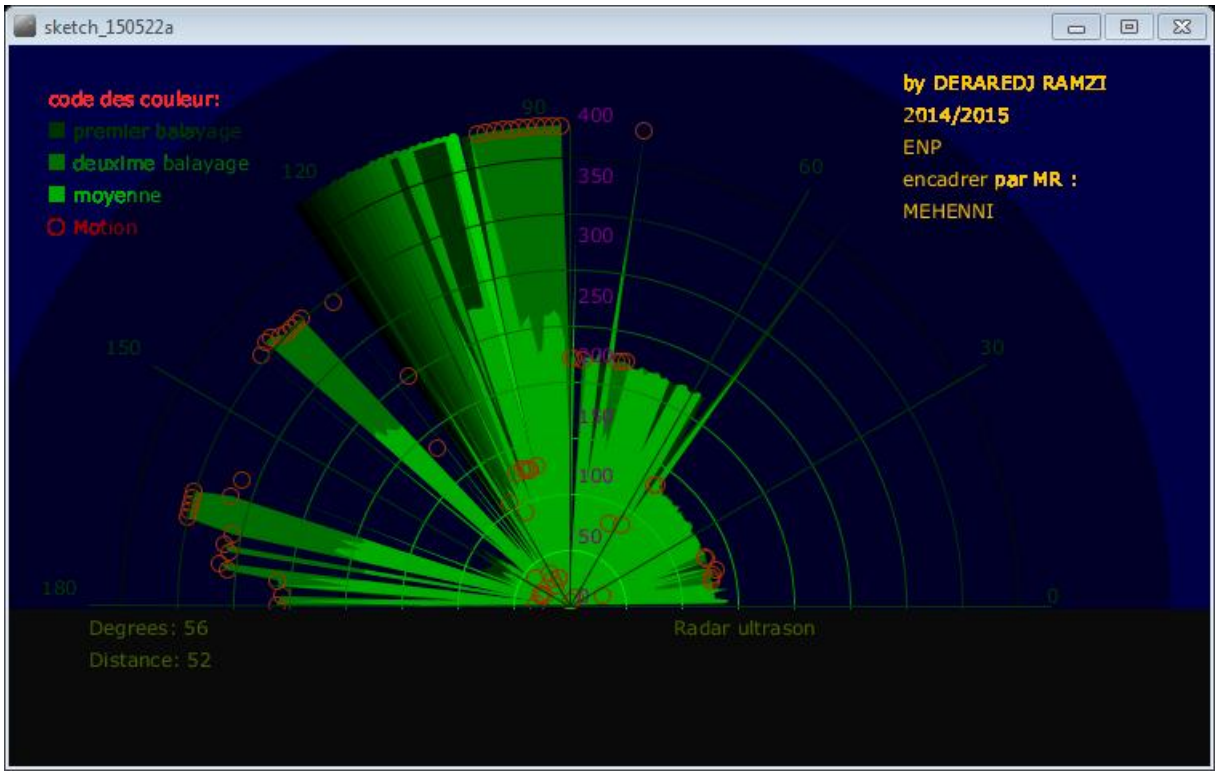


Figure 12: Résultat du code ci-dessus

4-CONCLUSION

Ce travail nous a permis de traiter une autre application de la télémétrie ultrasonore par la réalisation d'un radar. Notre radar qui réalise une opération appeler mapping en anglais et qui signifie la réalisation d'une cartographie de l'environnement avec lequel on interagit. Ce dernier peut permettre à un robot de se mouvoir dans l'espace cartographié avec notre radar grâce à la détection des cibles.

Bibliographie

[1] Eskimon et olyte ‘’ Arduino pour bien commencer en électronique et en programmation‘’ mars 2013 Jean-Noël Montagné ‘’ initiation à la mise en oeuvre matérielle et logicielle de l’Arduino’ ’novembre 2006

[2] R. Gregoire ‘’Processing’’ *septembre 2011*

[3] Datasheet HC-RS04

[4] Datasheet SR-03

[5] K. D. Schultz ’The trials and tribulations of building a phase-sensitive detector with an Arduino microcontroller’’ Department of Physics Hartwick College MAS-APS, 2014