LAB112

STAGE DE RECHERCHE EN LABORATOIRE

Projet final :

Conception d'un système d'autopilote pour un sous-marin à propulsion humaine

---

Auteure :

Martínez Gainza, Aurora

4 mai 2018

École de technologie superieure

Table des matières

TABLE DES MATIÈRES

Table des figures

Chapitre 1

Cahier des charges

1.1 Introduction

1.1.1 Client

Le client à qui s'adresse le projet est le club étudiant Omer de l'ÉTS. Il participe à chaque année à une compétition internationale d'ingénierie. La prochaine compétition est l'European International Submarine Races organisée par l'Institute of Marine Engineering, Science Technology (IMarEST). Les équipes doivent concevoir et construire des sous-marins et ensuite courser avec ceux-ci inondés d'eau dont le pilote, un plongeur, représente l'unique source de puissance. Aucune autre source d'accumulation d'énergie ne peut être utilisée. Omer, l'équipe de l'ETS, détient actuellement deux records mondiaux en vitesse dans plusieurs catégories de cette compétition soient monoplace sans hélice et bi-place à hélice. Chacun de leur sous-marin est propulsé par la seule force humaine comme indiqué dans les règlements du concours.

1.2 Présentation du projet

1.2.1 Contexte du domaine d'application

Le club étudiant Omer conçoit et construit des sous-marins à propulsion depuis plus de 25 ans. Les sous-marins sont remplis d'eau et leur puissance propulsive doit venir entièrement de l'effort du pilote à l'intérieur du sous-marin. Le club Omer est l'équipe à battre chaque année. Le club détient plusieurs records du monde et est champion du monde depuis plusieurs années.

Le club participe à deux types de compétition en alternance chaque année. La première, se déroulant aux États-Unis, est une course de vitesse en ligne droite. La seconde,

en Angleterre, est une course plus axée sur l'agilité et la vitesse avec un parcours comprenant une ligne droite, un demi-tour et un slalom.

1.2.2 Environnement du projet

Le système de direction actuel du sous-marin est électronique, mais manuel. Le pilote doit à la fois pédaler et diriger l'habitacle via un joystick, tant à l'horizontale qu'à la verticale. Un microcontrôleur permet par la suite de contrôler les quatre servomoteurs à l'arrière du sous-marin pour diriger celui-ci. Selon les conditions, il est parfois difficile pour le pilote de s'orienter dans les bassins de compétition malgré une ligne lumineuse et des bouées qui détermine le parcourt de la course. Ainsi, pour la prochaine version du sous-marin, Omer 11, il a été décidé d'implanter un système de contrôle automatisé de la direction. Omer 11 serait ainsi le premier sous-marin à propulsion humaine à avoir un tel système.

Le club travaille déjà à l'interne sur la partie verticale de la direction, soit le contrôle de profondeur. Toutefois, pour l'orientation horizontale, plusieurs avenues sont possibles et le club est en manque de ressource pour effectuer ce développement.

1.2.3 Définition de l'usage et l'utilisateur final

L'utilisateur final du projet va être le pilote du sous-marin. Jusqu'à présent, c'était lui qui devait diriger le véhicule avec le joystick. Comme cela a été mentionné avant, c'est lui qui s'occupe de la propulsion et la direction. Avec l'implémentation du système d'autopilote, il aura la possibilité de se concentrer sur la propulsion.

1.3 Les objectifs

1.3.1 Objectifs globaux

Développer un système d'orientation qui fournirait la position sur le plan horizontal, en coordonnées X et Y ainsi que l'angle du sous-marin par rapport à l'axe X à l'ordinateur de bord pour permettre de diriger le véhicule de façon autonome sans intervention du pilote. Trouver les capteurs nécessaires afin d'arriver à ces fins.

1.3.2 Objectifs spécifiques

— Trouver les capteurs nécessaires pour déterminer la position du sous-marin. Ces capteurs doivent pouvoir être logés dans le sous-marin sans nuire à l'ergonomie du

pilote.

- Développer un code capable d'obtenir les mesures de distance des capteurs
- Intégrer le système fabriqué avec l'ordinateur de bord Udoo Néo sous Linux Ubuntu

1.4 Les résultats attendus

1.4.1 Conditions minimales de satisfaction

Le système devra être capable de proportionner la position sur le plan horizontal, en coordonnées X et Y, avec une précision supérieure à 20 cm ainsi que l'angle du sous-marin par rapport à l'axe X, avec une précision supérieure à 5°.

1.4.2 Méthodologie de confirmation des résultats

Plusieurs test seront faites dans une piscine de dimensions connues avec la finalité de prouver le système. Le véhicule va être placé en positions ou les distances aux murs sont connues et les valeurs obtenues par les capteurs seront comparés a les vrais distances à fin de mesurer l'erreur et d'effectuer une possible calibration du système.

1.5 Charges de travail

1.5.1 Charges techniques

- Conception électrique comprenant les connexions des capteurs à la carte d'acquisition.
- Conception logicielle comprenant la programmation d'une librairie pour acquise les donnes des capteurs.
- Fabrication du système
- Tests du système et ajustements

1.5.2 Charges non techniques

- Approvisionnement/Achats des pièces sur divers site web
- Réalisation du cahier des charges énumérant les spécifications du système à concevoir
- Réalisation du rapport de conceptualisation

— Réalisation du rapport final

— Réalisation d'un manuel d'utilisation du système

— Réalisation de la présentation finale et démonstration du fonctionnement du système au client

1.6 Livrables

1.6.1 Livrables finaux

1. Le modèle du ou des capteurs choisis pour arriver à positionner le sous-marin. Ces capteurs peuvent être achetés (ou commandités) par le club s'ils sont choisis à temps.

2. Une librairie de codes écrit en C++ afin de recueillir les informations du ou des capteurs et ainsi positionner le sous-marin dans la piscine. Cette librairie doit pouvoir fonctionner avec l'ordinateur de bord, un Udoo Néo sous Ubuntu. Cette librairie doit pouvoir fournir la position en X, Y ainsi que l'angle du sous-marin par rapport à l'Axe X. Cette librairie doit pouvoir être facile d'utilisation et être facilement adaptable à différentes grosseurs de bassin.

3. Un rapport de projet expliquant clairement les étapes réalisées afin de parvenir au système choisi. Ce rapport doit pouvoir servir de guide pour les futurs membres du club afin de pouvoir bien comprendre le fonctionnement du système. Idéalement, ce rapport (guide) devrait pouvoir être assez compréhensible pour qu'une personne ayant peu de connaissance en programmation puisse faire fonctionner le système.

1.6.2 Livrables temporaires

Élement	**Date**
Cahier des charges (CdC)	22/01/2018
Rapport de conceptualisation (RdC)	05/02/2018
Capteurs et circuits électriques	26/02/2018
Librairie en C++	26/03/2018

TABLE 1.1 – Livrables temporaires

1.7 Le plan de projet

1.7.1 Phases

Afin de bien mener à terme le projet, celui-ci sera divisé en plusieurs phases soit de conception, de fabrication, de développement et de tests. C'est important aussi bien déterminer la durée assignée à chacune des phases, ainsi que l'ordre dans lequel elles seront effectuées, dans le but de faire une planification efficace du temps disponible pour faire le projet.

1.7.2 Phases du projet

Phase	Détails	Début	Fin
Conceptualisation	Conception d'une solution adaptée aux besoins. Plusieurs systèmes seront étudiés avant choisir quel est le meilleur	22/01/2018	4/02/2018
Réalisation matérielle	Chercher entre les différents capteurs disponibles sur le marché lesquels sont les plus adéquat pour le problème, avec les présentes restrictions (résistance à l'eau à 10m de profondeur, grandeur réduite)	6/02/2018	25/02/2018
Réalisation logicielle	Développement d'une librairie de codes écrit en C++ afin de recueillir les informations des capteurs qui fonctionnent avec l'ordinateur de bord.	27/02/2018	25/03/2018
Tests et documentation	Effectuer les tests nécessaires pour vérifier le bon fonctionnement du projet dans le futur et corriger les possibles erreurs. À la fin, écrire un rapport de projet expliquant clairement les étapes réalisées qui doit servir de guide pour les futurs membres du club afin de pouvoir bien comprendre le fonctionnement du système	27/03/2018	1/05/2018

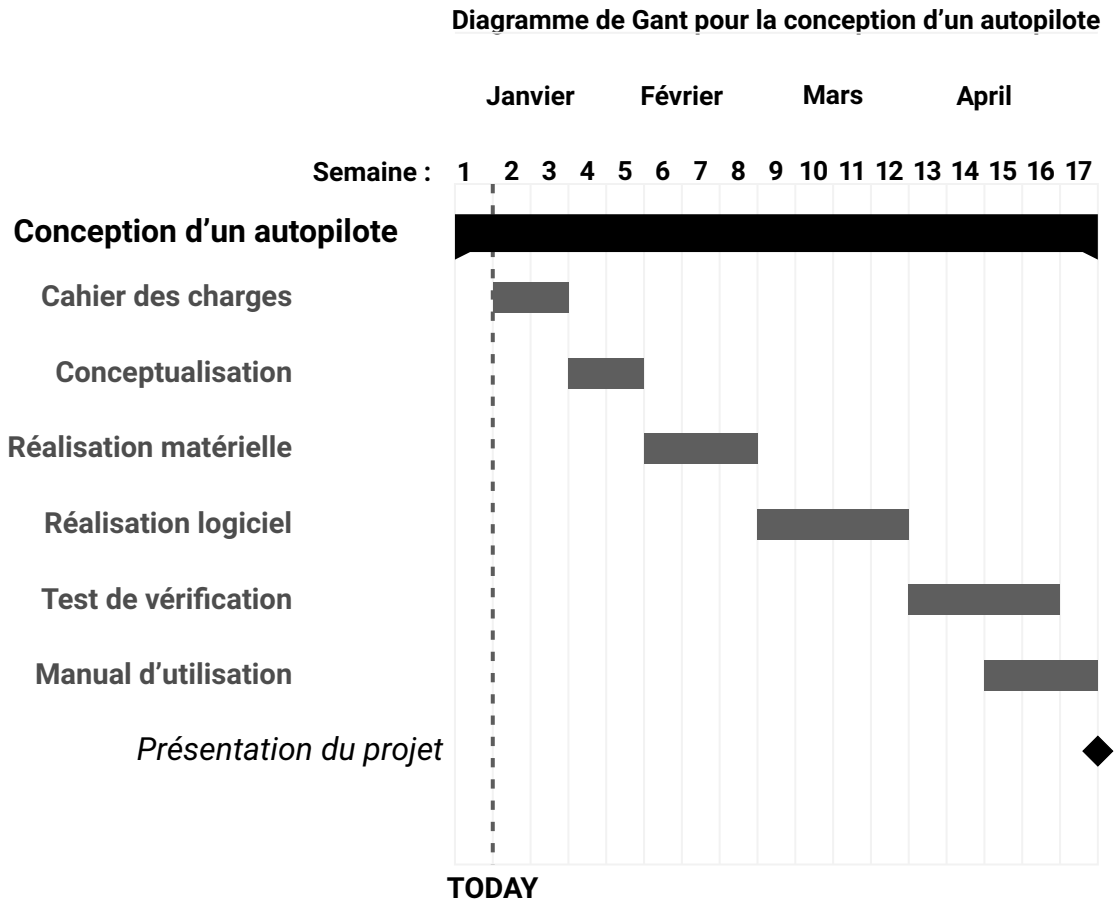
TABLE 1.2 – Phases du projet

1.7.3 Dates importantes

Élement	Date
Cahier des charges (CdC)	22/01/2018
Rapport de conceptualisation (RdC)	05/02/2018
Prototype fonctionnel pour la présentation finale	02/05/2018
Rapport final	02/05/2018
Présentation finale devant le client	02/05/2018

TABLE 1.3 – Tableau des dates importantes

1.7.4 Diagramme de Gant



1.8 Les risques et les opportunités

Probabilité d'occurrence	Impact sur le projet	Poids du risque
1 = Faible 2 = Moyenne 3 = Forte 4 = Très forte	1 = Mineur 2 = Moyen 3 = Important 4 = Majeur	Poids du risque (PdR) = Probabilité d'occurrence x Impact sur le projet

TABLE 1.4 – Niveaux possibles de probabilité d'occurrence et d'impact sur le projet

1.8.1 Opportunités

Identification	**Probabilité**	**Impact**	**Avantage**
Temps de développement du logiciel pour contrôler les capteurs plus court que prévu	1	4	Plus de temps pour les prochaines phases de le projet
Obtention des bons résultats au premier test fait	2	3	Possibilité d'améliorer le système à fin de le faire plus efficace

TABLE 1.5 – Tableu des opportunités

1.8.2 Risques

Identification	**Probabilité**	**Impact**	**PdR**	**Action**
Retard avec la livraison des capteurs commandés	2	3	6	Essayer de trouver un capteur équivalentes de façon local.
Dysfonction des composantes	1	3	3	S'assurer de que les capteurs choisies peuvent être commandées rapidement.
Logiciel dans le système non-fonctionnel	2	4	8	Chercher l'aide d'autres personnes qui ont de l'expérience dans la matière.

TABLE 1.6 – Tableau des risques

1.9 Hypothèses de départ et contraintes

— Le système d'orientation doit s'imbriquer aisément avec le système de contrôle actuel :

  - Ordinateur de bord Udoo Néo sous Linux Ubuntu
  - Tension d'alimentation de 11.4v

— Le système doit pouvoir être adaptable aux deux compétitions. Celle d'Angleterre est un bassin rectangulaire d'environ 125m par 75m et celle aux États-Unis est dans un bassin d'une largeur d'environ 50m de large et la course se déroule sur une distance d'environ 100m.

— Le système doit être étanche jusqu'à 10 mètres de profondeur. Des boîtiers pourraient toutefois être fabriqués pour contenir le système choisi.

— L'ensemble du projet doit respecter le budget de 750$, les composantes peuvent toutefois provenir de commandites pour le club.

1.10 Conclusion

Ce cahier des charges a été conçu pour résumer les besoins du client et bien spécifier les objectifs du projet, les hypothèses de départ et les contraintes, avant de commencer la conceptualisation du système. Les différentes phases du projet ont été planifiées, ainsi que le temps investi dans chacune d'elles, avec le but de faire une bonne gestion du temps disponible. Afin de mieux prévoir les délais qui pourraient survenir en cours de projet, les risques et les opportunités ont été analysés pour éviter toutes les adversités possibles.

Chapitre 2

Rapport de conceptualisation

2.1 Introduction

2.1.1 Objectives

Les objectives de le rapport de conceptualisation sont les suivants :

— Faire la recherche des différents modèles fonctionnels qui peut être une solution au problème originel.

— Étudier les différentes caractéristiques de chacun des modèles à fin de trouver lequel est plus proche des spécifications techniques requises.

— Développer le concept du modèle choisi avant commencer sa construction.

2.2 Génération et recherche d'idées de fonctionnalités

2.2.1 Spécifications requises

Au début, il faut rappeler les spécifications que les solutions à chercher doivent suivre :

— Le système d'orientation doit s'imbriquer aisément avec le système de contrôle actuel :

  • Ordinateur de bord Udoo Néo sous Linux Ubuntu

  • Tension d'alimentation de 11.4v

— Le système doit pouvoir être adaptable aux deux compétitions. Celle d'Angleterre est un bassin rectangulaire d'environ 125m par 75m et celle aux États-Unis est dans un bassin d'une largeur d'environ 50m de large et la course se déroule sur une distance d'environ 100m.

— Le système doit être étanche jusqu'à 10 mètres de profondeur. Des boîtiers pourraient toutefois être fabriqués pour contenir le système choisi.

— L'ensemble du projet doit respecter le budget de 750$, les composantes peuvent toutefois provenir de commandites pour le club.

2.2.2 Recherche des idées

À fin de trouver des idées fonctionnels qui suivent les spécification décrites avant, on a observe les méthodes utilisées pour autres véhicules similaires pour connaître sa position, à fin d'étudier si c'est possible les adapter à nôtre problème.

Cette recherche à été faite d'un côte sur l'internet et d'autre côte en demandant à experts dans la matière, comme professeurs ou membres d'autres clubs étudiants de l'ÉTS qui ont déjà implémenté des systèmes similaires.

2.3 Élaboration des concepts de solution

2.3.1 Présentation des différents concepts

Les principales concepts de solution auxquels on a arrivé pour connaître la position de le véhicule sont les suivants :

— **IMU + GPS** Avec une Unité de Mesure Inertiel c'est possible connaître les changes de vitesse et orientation, mais ça induit un erreur que doit être corrigé systématiquement par un capteur de GPS.

— **Sonar/lidar** Cela consiste en placer des capteurs d'ultrasons (sonar) ou laser(lidar) sur le véhicule. Cet système permettre connaître la position du véhicule par rapport au murs du bassin.

— **Système de caméras** Avec une caméra c'est possible filmer le parcours et après connaître la position par identification des images.

2.3.2 Tableau d'avantages et inconvénients de chacun des systèmes

	Avantages	Inconvénients
IMU+GPS	-Permet connaître la position d'un façon facile -En intégrer les résultats de l'IMU permet avoir une bonne précision	-Mauvais fonctionnement avec une profondeur supérieure à 50 cm -Dépends de l'obtention d'une bonne signal du GPS
Sonar/lidar	-Bonne fonctionnement dans d'un bassin avec des murs -Simple à programmer	-Le prix surpasse le budget disponible -Si le prix surpasse pas le budget, la distance de mesure n'est pas la correcte -Possibles interférences avec d'autres objets dans le bassin
Vision	-Bonne fonctionnement dans un parcours connu -Marche pour n'importe quel parcours, n'a pas besoin des murs.	-Difficile à programmer -Sensible à interférences lumineuses dans l'eau

TABLE 2.1 – Avantages et inconvénients

2.4 Analyse de faisabilité des idées et des concepts

2.4.1 Méthode à utiliser pour faire l'analyse

Une table avec les différentes caractéristiques des systèmes à étudier a été élaboré à fin d'analyser la fonctionnalité des concepts présentés antérieurement. Un pourcentage á été attribué a chaque caractéristique en fonction de son importance pour la viabilité de chacun des concepts. Chaque évaluation à été faite de 0 a 4, ou 0 représentes que le système réponds pas à les spécifications de design et 4 que le système suive le spécification parfaitement.

2.4.2 Tableau d'analyse de faisabilité

Systèmes	Ponderation [%]	**IMU+GPS**	**Sonar/Lidar**	**Vision**
Fonctionabilitédans l'eau	50	0	4	3
Prix	20	4	1	4
Facilité d'assemblage	10	4	2	3
Facilité de programmation	5	3	4	2
Qualité des résultats	15	3	4	3
Total		220	290	**325**

TABLE 2.2 – Comparaison des différents systèmes

2.4.3 Résultat de l'analyse

Après l'analyse, le concept le plus adéquate à les spécifications d'origine c'est le système de vision. Le système de sonar/lidar est aussi une bonne solution à considérer, puisque son plus gros inconvénient est le prix. C'est possible refuser complètement le système de IMU+GPS à cause de que les ondes du GPS rentrent pas dans l'eau a une profondeur supérieure à 50 cm.

2.5 Présentation du concept retenu

2.5.1 Principe de fonctionnement

Le concept choisi finalement était la vision par ordinateur. Pour implémenter ce système la, c'est nécessaire une caméra connecté à la carte Udoo déjà présente au véhicule. La caméra doit filmer une fois tout le parcours, et enregistrer le vidéo. Après, elle peux connaître sa position dans le même parcours par identification des images. Cela veut dire qu'elle fait une comparaison en temps réelle entre l'image actuel et les images déjà enregistrées, et voire laquelle est la plus pareille.

2.5.2 Diagramme fonctionnel

Avec le suivant diagramme fonctionnel c'est possible apprécier le schéma général du système. La caméra est bronché directement à la carte Udoo qui gouverne le système. La-bas, les images sont comparés avec les images enregistrés antérieurement, à fin de déterminer la position en X et Y du véhicule. La-bas, avec un double PID, la position réelle est comparé à la position requise et cet information est transfère à la carte Arduino Nano responsable de contrôler l'angle des servos qui règlent la position des ailerons à l'arrière du véhicule.

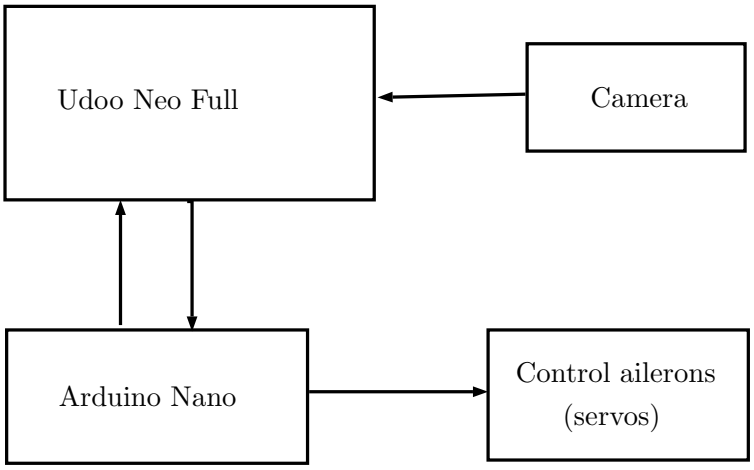


FIGURE 2.1 – Diagramme des bloques explicative du système

2.5.3 Hardware

Le hardware à utiliser pour le projet c'est une caméra résistante à l'eau connecté a une carte d'acquisition. La caméra et la carte à utiliser sont les suivants :

- **Carte :** Udoo Neo Full
  https ://shop.udoo.org/usa/udoo-neo-full1.html
- **Caméra :** Camera Analogique pour Lumière Faible
  https ://www.robotshop.com/ca/fr/caméra-analogique-lumiere-faible.html
  Leur caractéristiques techniques pour lesquelles elle a été sélectionné sont les suivants :
  - Peut être utilisée dans des applications sous-marines
  - Sensibilité à la lumière de seulement 0,0003 lux
  - Objectif de 2,1 mm
  - Comprend le capteur CCD Super HAD 810 1/3 " de Sony

2.5.4 Logiciel

Les logiciels à utiliser son les suivants :

- **Linux Ubuntu**
  C'est le système opératif avec qui travaille la carte Udoo.
- **Open CV** (Open Source Computer Vision Library)
  Pour travailler avec le traitement des images, les librairies à utiliser sont Open CV. Le langage ce programmation est C++, compatible avec Linux.

Remarquer que tous les logiciels à utiliser sont open source, gratuites pour tous les utilisateurs.

2.6 Conclusion

Ce rapport de conceptualisation a été conçu pour résumer les différents concepts de solution étudies, ainsi que les avantages et inconvénients de chacun d'eux. Un analyse de faisabilité à été fait afin de choisir la solution plus proche aux spécifications initiales. Cette solution à été présente en détaille, son diagramme fonctionnel, le hardware et les logiciels à utiliser pendant l'implémentation de la même.

Chapitre 3

Cahier de réalisation

3.1 Introduction

Le projet présente consiste à trouver une solution pour connaître la position d'un sous-marin dans une piscine. L'objectif initial est être capable de trouver la position en x, y et l'angle horizontal.

Dans le cahier de réalisation, les solutions à effectuer sont présentées, pour expliquer après son fonctionnement et son utilisation. Serai explique aussi la réalisation physique de la solution ainsi que les limites techniques du système.

3.2 Présentation des solutions

À fin de donner une solution efficace au problème, on à effectué deux de les solutions possibles : elle de vision par ordinateur et elle avec des ultrasons.

La solution avec des ultrasons consiste à installer deux capteurs d'ultrasons dans le latérale du véhicule a une distance *d* entre eux. De cette façon, c'est possible mesurer la distance par rapport au mur et l'angle α, comme c'est possible apprécier dans l'image suivant, ou *x1* et *x2* sont les distances mesurées pour chacun des capteurs d'ultrasons.

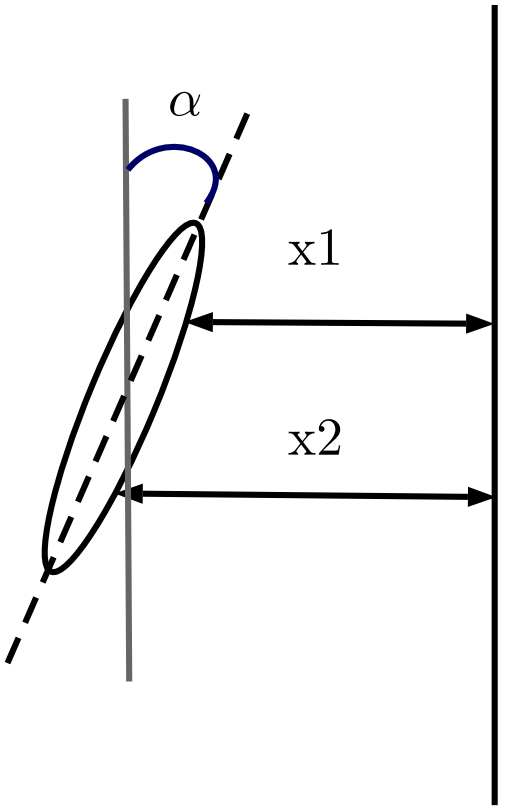


FIGURE 3.1 – Schéma de la solution des ultrasons

La solution avec la vision consiste à installer une caméra qui filme en temps réel le fond de la piscine. Au fond, il y a une ligne pour guider la trajectoire. Le but du programme réalise c'est de suivre cette ligne.

3.3 Description technique du prototype

À continuation on va exposer les caractéristiques des capteurs choisis pour la réalisation du projet.

3.3.1 Système d'ultrasons

Les spécifications techniques des capteurs d'ultrasons sont les suivants :

- **Range détectable entre 25cm et 450cm :** à cause de cette caractéristique c'est possible mesurer la distance seulement par rapport au mur latéral.
- **Résistante à l'eau :** cela était indispensable puisque on va l'installer dans un véhicule sous-marine.
- **Résolution de 0.5 cm :** c'est adéquate avec les prestations requises.

— **Tension d'entrée : 5 V DC :** parfait pour bien marcher avec la carte UDOO laquelle à une sortie à 5V.

FIGURE 3.2 – Capteur d'ultrasons utilisé

3.3.2 Système de vision

Les spécifications techniques de la caméra utilisé sont les suivants :

— Peut être utilisée dans des applications sous-marines

— Sensibilité à la lumière de seulement 0,0003 lux

— Objectif de 2,1 mm

— Comprend le capteur CCD Super HAD 810 1/3 " de Sony

FIGURE 3.3 – Caméra analogique pour lumière faible

3.4 Fonctionnement

3.4.1 Système d'ultrasons

Pour programmer les capteurs d'ultrasons on a utilisé le module arduino présente dans la carte UDOO.

À continuation on va expliquer le code utilisé pour calculer la distance et l'angle avec le système d'ultrasons. Au début, il faut définir les pins de la carte qu'on va utiliser comme émetteurs *(TriggerPin)* de la signal d'ultrasons, et eux qu'on va utiliser comme récepteurs *(EchoPin)* :

```
const int EchoPin = 10;
const int TriggerPin = 9;
const int EchoPin2 = 8;
const int TriggerPin2 = 7;
const int LedPin = 13;
```

FIGURE 3.4 – Définition des pins

Les pins définies comme émetteurs il faut les configurer comme output (sorties). Au même temps, ils considérés comme récepteurs il faut les configurer comme inputs (entrées) :

```
void setup() {
   Serial.begin(19200);
   pinMode(LedPin, OUTPUT);
   pinMode(TriggerPin, OUTPUT);
   pinMode(EchoPin, INPUT);
   pinMode(TriggerPin2, OUTPUT);
   pinMode(EchoPin2, INPUT);
}
```

FIGURE 3.5 – Configuration des pins

Dans la boucle principal on utilise la fonction *ping*, expliqué plus tard, pour calculer la distance mesuré pour chaque ultrason, ainsi que la fonction *CalcAngle* pour calculer l'angle. À continuation les valeurs obtenues sont imprimés sur la fenêtre de commandes avec la fonction *serial.print*.

```
void loop() {
   int x1 = ping(TriggerPin, EchoPin);
   int x2 = ping(TriggerPin2, EchoPin2);
   double x = x1-x2;
   double y = 100; //distance entre les capteurs,1 m
   int angle = CalcAngle( x, y);
   Serial.print("x1: ");
   Serial.println(x1);
   Serial.print(" ");
   Serial.print("x2: ");
   Serial.println(x2);
   Serial.print("Angle:");
   Serial.println(angle);

   delay(1000);
}
```

FIGURE 3.6 – Void loop

La fonction *ping* mesure la distance en cm en calculant le temps de la signal émis en faire la trajet d'allé-retour. Un pulse d'une durée de 10 micro-seconds est génère chaque 4 micro-seconds.

```
int ping(int TriggerPin, int EchoPin) {
long duration, distanceCm;
digitalWrite(TriggerPin, LOW);  //pour générer un pulse bien défini on fixe LOW à 4us
delayMicroseconds(4);
digitalWrite(TriggerPin, HIGH);  //generation Trigger (disparo) de 10us
delayMicroseconds(10);
digitalWrite(TriggerPin, LOW);
duration = pulseIn(EchoPin, HIGH);  //on mesure le temps entre pulses, en microsecondes
distanceCm = duration * 10 / 292/ 2;   //conversion de distance, en cm
return distanceCm;
}
```

FIGURE 3.7 – Fonction pour mesurer la distance

La fonction *CalcAngle* calcule l'angle de la façon suivant :

$$\alpha = ArcTan\left(\frac{x1 - x2}{y}\right)$$

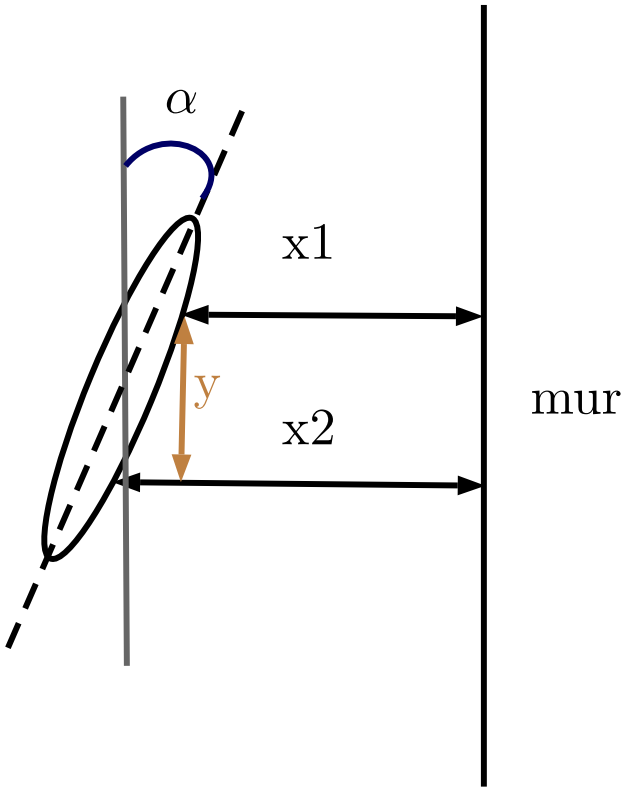


FIGURE 3.8 – Schéma ArcTan

Pour utiliser en arduino la fonction *atan2* c'est nécessaire d'inclure la librairie *math.h*.

```
int CalcAngle(double x, double y){ //fonction pour calculer l'angle
float Pi = 3.14159;
float angle =  (atan2 ( x, y)*180)/Pi; // arc tangent of y/x
return angle;}
```

FIGURE 3.9 – Fonction pour calculer l'angle

A continuation on observe que les deux distances et l'angle apparaît dans le terminal linux de l'UDOO après l'instruction :

minicom -D /dev/ttyMCC

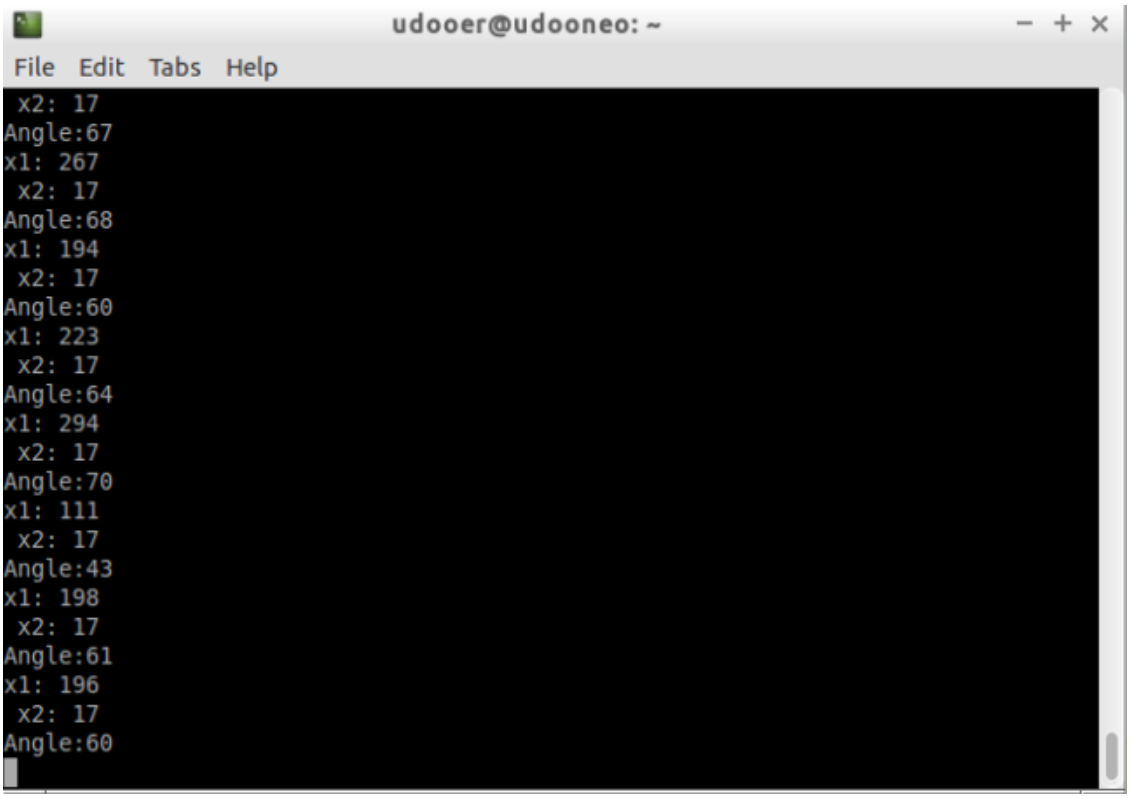


FIGURE 3.10 – Terminal

3.4.2 Système de vision

Pour programmer le système de vision qui détecte la ligne on a employé les librairies de OpenCV, avec un code en c++. À fin de détecter la ligne, on a utilisé des algorithmes de détection de contours. Les coordonnés des points du contour son enregistrés dans deux vecteurs (un pour chaque côté de la ligne).

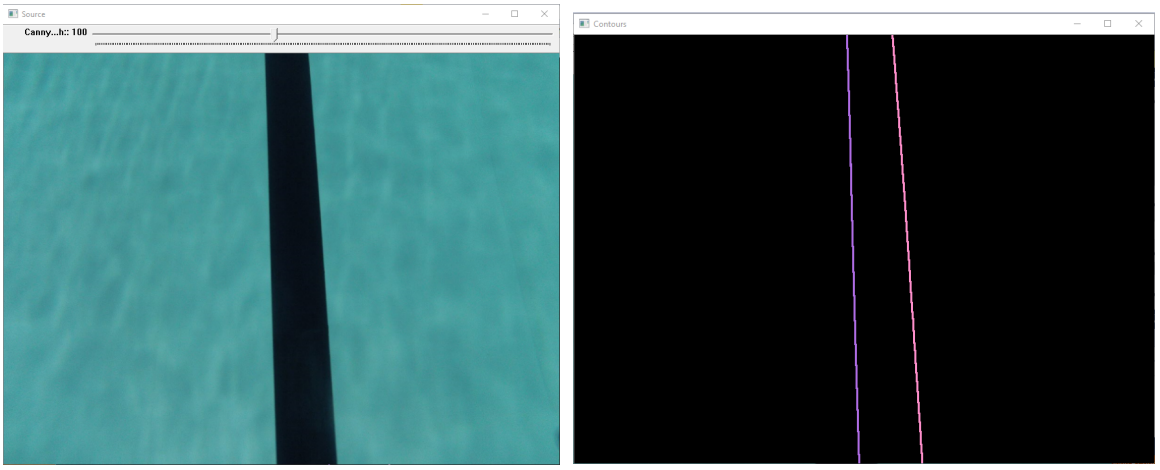


Image originel Contours détectés

FIGURE 3.11 – Détection des contours

Pour connaître la position en x par rapport a la ligne, on fait la différence entre la position en x du premier pixel du vecteur (lui avec y=0) et la coordonné x du pixel au milieu de l'image. Pour calculer l'angle, on calcule l'arctangent du premier et dernier pixel de chaque vecteur (chaque contour) et on fait la moyenne.

3.5 Réalisation

Dans ce point la on va expliquer la réalisation physique ainsi que la communication avec l'ordinateur de notre système d'ultrasons.

3.5.1 Réalisation physique

Matériel employé :

- Breadboard
- Carte UDOO
- 2 capteurs d'ultrasons
- Ordinateur
- Fil micro USB
- Fils électriques unifilaires

Pour l'assemblage du système on a bronché les deux capteurs d'ultrason a la carte UDOO avec l'aide d'un breadboard. Chaque capteur a quatre pins : Echo, Trigger, Ground et alimentation a 5V. Le ground et l'alimentation à 5V sont connectés aux pins de l'UDOO

définis pour ces fonctions. Le Trigger est connecté à un pin qu'on a défini comme output et le Echo, a un défini comme input. La carte UDOO est alimenté à 5V avec l'ordinateur par milieu du fil micro USB. À continuation on observe l'assemblage en fonctionnement :

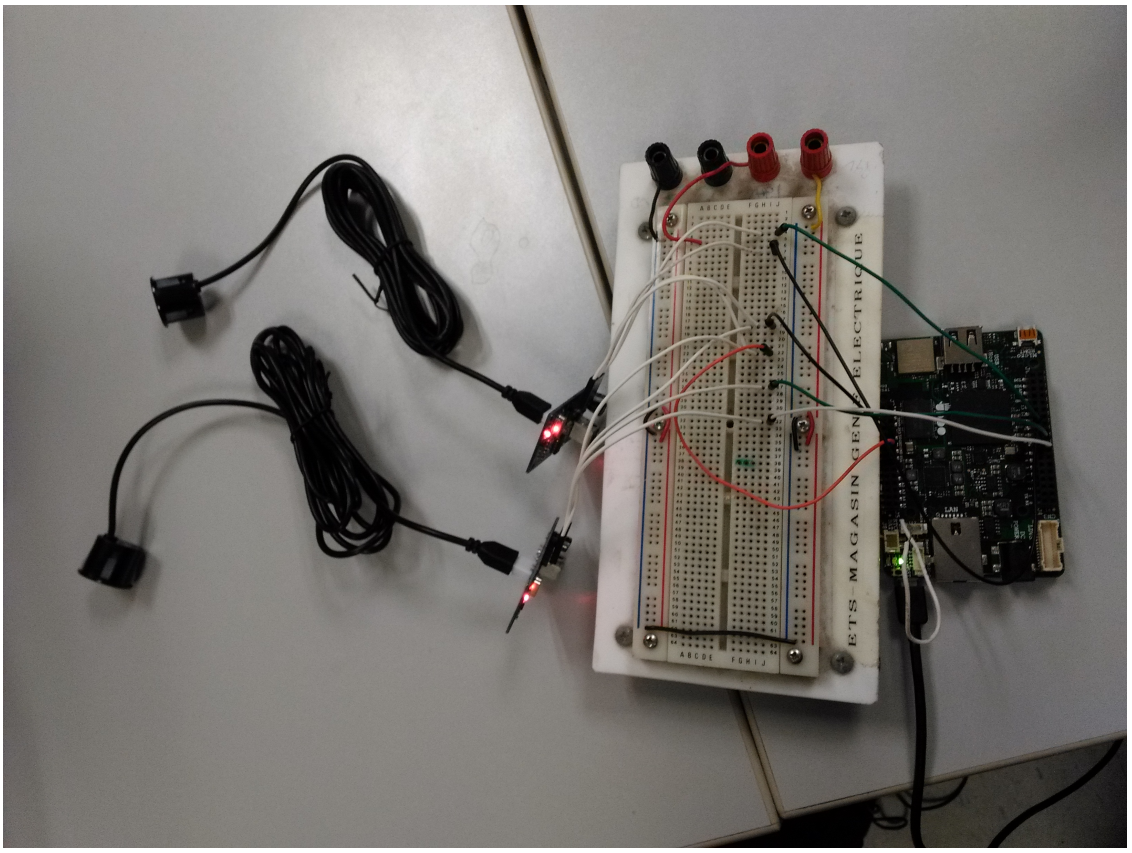


FIGURE 3.12 – Basse de données

3.5.2 Communication avec l'ordinateur

Pour faire la communication, on a bronché la carte UDOO à l'ordinateur avec un fil micro USB. Avec le logiciel VNC nous sommes capables d'ouvrir le système d'exploitation linux ubuntu de la carte. Pour exécuter le programme, il faut ouvrir l'arduino dans linux. Une fois le programme marche, sur la fenêtre terminal il faut écrire l'instruction :

```
minicom -D /dev/ttyMCC
```

Après cela, c'est possible observer la distance et l'angle qui apparaît à l'écran en temps réel.

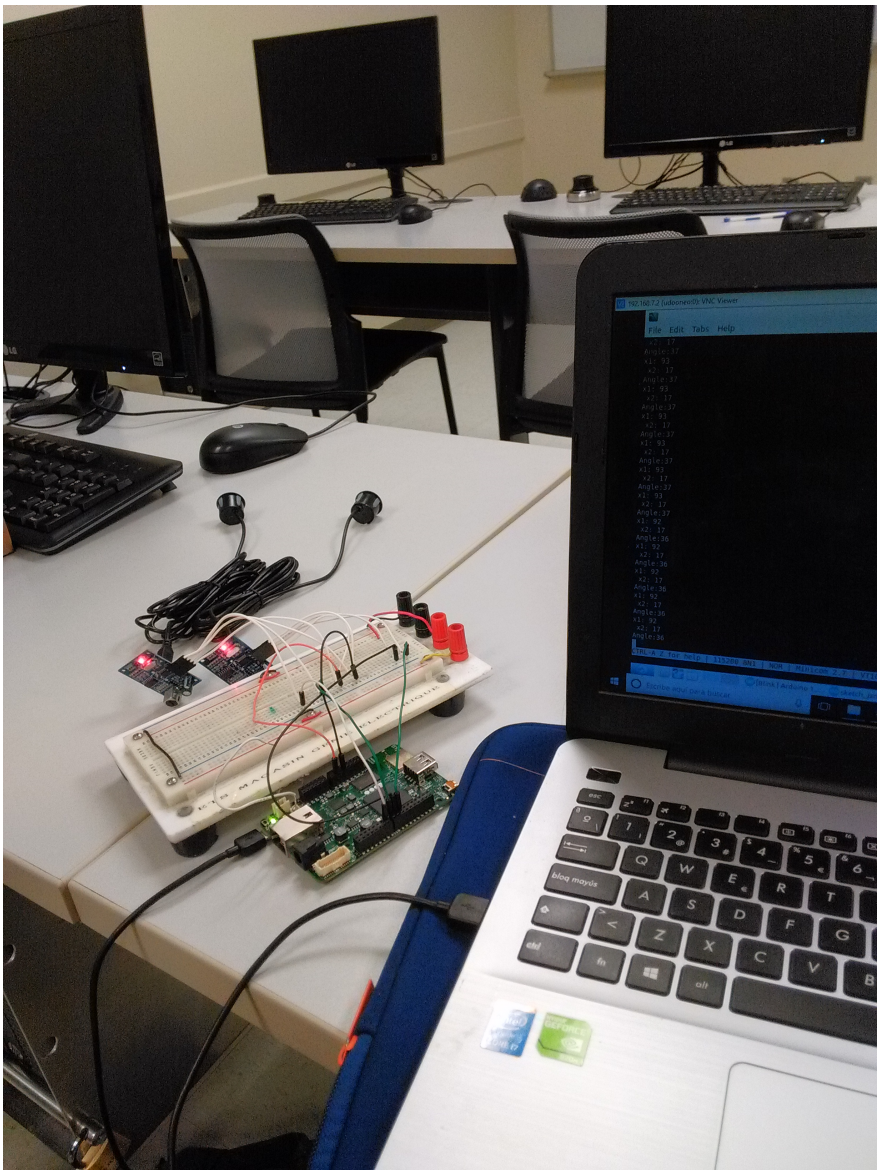


FIGURE 3.13 – Réalisation physique

3.6 Performance et limites du système

Après effectuer le test du système, les limitations trouvés sont les suivants :

- Quand le mur est pas proche, il proportion des mesures aléatoires qui peuvent devenir mélangent avec des vrais mesures.
- Comme sa distance de mesure maximal c'est de 4.5 m il est pas capable de mesurer la distance longitudinal en y.
- Le système présente un delay d'environ 0.5 s.

3.7 Travaux futurs et problèmes connus

3.7.1 Problèmes connus

Les principales problèmes qu'on a trouvé pendant la réalisation du projet sont les suivants :

1. Système de vision
    - Difficulté à broncher la caméra à la carte d'acquisition UDOO, à cause de qu'elle proportionne une signal analogique.
    - Difficulté à suivre la ligne quand il y a d'autres objets qui apparaît dans le champ de vision de la caméra.
    - Cet système c'est capable de suivre la ligne mais il est pas capable de proportionner la position en x, y et l'angle du véhicule.
2. Système d'ultrasons
    - Il proportionne la position en x par rapport au mur latéral et l'angle mais il proportionne pas la position en y par rapport au mur frontal.
    - Avec les ultrasons utilisés c'est pas possible mesurer la distance quand le mur est plus loin de 5 m.

3.7.2 Travaux futurs

1. Système de vision
    - Améliorer le code à fin qu'il détecte pas d'autres objets que la ligne à suivre.
    - Ajouter d'autres caméras pour améliorer la précision du système.
2. Système d'ultrasons
    - Utiliser d'autres capteurs capables de mesurer une distance plus grand (environ 50 m).
    - Ajouter un troisième capteur à fin de mesurer aussi la distance en y.

3.8 Conclusion

Dans ce projet, on a trouvé des solutions pour le problème du positionnement d'un sous-marine à propulsion humaine dans une piscine. Les solutions trouvées sont, d'un côté, un système d'ultrasons, et d'autre côté, un système de vision par ordinateur avec une caméra.

Toutes les deux solutions ont été réalisés, et finalement à était choisi elle des ultrasons, grâce a sa simplicité et sa bonne performance. Avec cette solution, c'est possible mesurer la distance en x par rapport au mur ainsi que l'angle horizontal.

Comme travaux futurs, existe la possibilité d'acheter d'autres capteurs capables de mesurer des distances plus grandes, et ajouter un troisième, pour mesurer la position en y aussi.

Appendices

.1 Code c++ OpenCV

```
#include "opencv2/imgcodecs.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
using namespace cv;
using namespace std;
Mat src; Mat src_gray;
int thresh = 100;
int max_thresh = 255;
RNG rng(12345);
void thresh_callback(int, void*);
int main(int, char** argv)
{
  //VideoCapture cap("video_piscina.mp4");
  //cap.retrieve(src, CV_CAP_OPENNI_BGR_IMAGE);

  src = imread("piscina.png");
  if (src.empty())
  {
    cerr << "No image supplied ..." << endl;
    return -1;
  }
  cvtColor(src, src_gray, COLOR_BGR2GRAY);
  blur(src_gray, src_gray, Size(3, 3));
  const char* source_window = "Source";
  namedWindow(source_window, WINDOW_AUTOSIZE);
  imshow(source_window, src);
  createTrackbar(" Canny thresh:", "Source", &thresh, max_thresh,
   thresh_callback);
  thresh_callback(0, 0);
  waitKey(0);
  return(0);


}
void thresh_callback(int, void*)
{
  Mat canny_output;
  vector<vector<Point> > contours;
  vector<Vec4i> hierarchy;
  //vector<Point> fifthcontour = contours.at(4);
```

```
  Canny(src_gray, canny_output, thresh, thresh * 2, 3);
45 findContours(canny_output, contours, hierarchy, RETR_TREE,
    CHAIN_APPROX_SIMPLE, Point(0, 0));

47 vector<int> ptsX;
   vector<int> ptsY;
49   for (int j = 0; j < contours.size(); j++) {
       Point ptsTemp;
51
       cout << "Ligne " << j << endl;
53     cout << contours[j][0] << endl;
       ptsTemp = contours[j][0];
55     ptsX.push_back(ptsTemp.x);
       ptsY.push_back(ptsTemp.y);
57     vector<Point>temp = contours[j];
       int taille = temp.size();
59     cout << contours[j][(taille-1)/2] << endl;

61     ptsTemp = contours[j][(taille - 1) / 2];
       ptsX.push_back(ptsTemp.x);
63     ptsY.push_back(ptsTemp.y);

65   }
     for (int k = 0; k < ptsX.size(); k++) {
67     cout << ptsX[k] << " " << endl;
     }
69   cout << atan((ptsX[3] - ptsX[2])/ (ptsY[3] - ptsY[2])) * 180 / 3.14159 <<
    " " << endl;
     for (int k = 0; k < ptsY.size(); k++) {
71     cout << ptsY[k] << " " << endl;
     }
73   int param1 = ptsX[3] - ptsX[2];
     int param2 = (ptsY[3] - ptsY[2]);
75   float param = param1/param2;
     float angle = atan(param) * 180 /3.14159 ;
77   cout << angle << endl;
     printf("The arc tangent of %f %f %f is %f degrees\n",param1,param2, param,
     angle);
79
   /*for (int j = 0; j < contours.size(); j++) {
81   cout << "Ligne " << j << endl;
     cout << contours[j] << endl;
83 }
   */
85
   //cout << cdfMax << endl;
```

```
87    //remplazar un pixel
      //for (int i = 0; i < src.rows; i++)
89    //{
      //  for (int j = 0; j < src.cols; j++)
91    //  {
      //    Vec3b bgrPixel = foo.at<Vec3b>(i, j);
93
      // do something with BGR values...  //  }
95    //}

97    Mat drawing = Mat::zeros(canny_output.size(), CV_8UC3);
      //std::cout << contours.size() << std::endl;
99    for (size_t i = 0; i< contours.size(); i++)
      {
101     Scalar color = Scalar(rng.uniform(0, 255), rng.uniform(0, 255), rng.
       uniform(0, 255));
       drawContours(drawing, contours, (int)i, color, 2, 8, hierarchy, 0, Point()
       );
103   }

105   //for (int i = 0; i < fifthcontour.size(); i++) {
      //  Point coordinate_i_ofcontour = fifthcontour.size();
107   //  cout << endl << "contour with coordinates: x = " <<
       coordinate_i_ofcontour.x << " y = " << coordinate_i_ofcontour.y;
      //}
109

      //FILE *outfile = fopen("coordinates.txt", "a+");
111   //fprintf(outfile, "%d\t%d\n", coordinate_i_ofcontour.x,
       coordinate_i_ofcontour.y);
      //fclose(outfile);
113   namedWindow("Contours", WINDOW_AUTOSIZE);
      imshow("Contours", drawing);
115 }
```

Code 1 – Code c++ OpenCV

.2 Code arduino ultrasons

```
#include <math.h>


const int EchoPin = 10;
const int TriggerPin = 9;
const int EchoPin2 = 8;
const int TriggerPin2 = 7;
const int LedPin = 13;

void setup() {
   Serial.begin(19200);
   pinMode(LedPin, OUTPUT);
   pinMode(TriggerPin, OUTPUT);
   pinMode(EchoPin, INPUT);
   pinMode(TriggerPin2, OUTPUT);
   pinMode(EchoPin2, INPUT);
  // pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
   int x1 = ping(TriggerPin, EchoPin);
   int x2 = ping(TriggerPin2, EchoPin2);
   double x = x1-x2;
   double y = 100; //distancia entre los dos sensores
   int angle = CalcAngle( x, y);
   Serial.print("x1: ");
   Serial.println(x1);
   Serial.print(" ");
   Serial.print("x2: ");
   Serial.println(x2);
   Serial.print("Angulo:");
   Serial.println(angle);

   delay(1000);
}

   int ping(int TriggerPin, int EchoPin) {
   long duration, distanceCm;
   digitalWrite(TriggerPin, LOW);  //para generar un pulso limpio ponemos a
   LOW 4us
   delayMicroseconds(4);
   digitalWrite(TriggerPin, HIGH);  //generamos Trigger (disparo) de 10us
   delayMicroseconds(10);
   digitalWrite(TriggerPin, LOW);
```

```
    duration = pulseIn(EchoPin, HIGH);  //medimos el tiempo entre pulsos, en
    microsegundos
45  distanceCm = duration * 10 / 292/ 2;   //convertimos a distancia, en cm
    return distanceCm;
47  }

49  int CalcAngle(double x, double y){ //funcion que calcula el angulo
    float Pi = 3.14159;
51  float angle =  (atan2 ( x, y)*180)/Pi; // arc tangent of y/x
    return angle;}
```

Code 2 – Code Arduino Ultrasons