



NAUTILUS - RAPPORT

Dylan Bideau, Julien Turpin, Pierre Bogrand, Guillaume Vincenti

10 Avril 2018

Sommaire

1	Introduction	2
2	Présentation du projet	3
3	Cahier des charges	4
3.1	Analyse Fonctionnelle	4
3.1.1	Structure	4
3.1.2	Commandabilité	4
3.1.3	Milieu d'utilisation	4
3.1.4	Energie	5
3.1.5	Motorisation	5
3.1.6	Acquisitions	5
4	Acquisition et Commandabilité	6
4.1	Raspberry	6
4.1.1	Cameras	6
4.1.2	Capteur de Pression/Temperature	9
4.1.3	Central Inertielle	9
4.2	PC	10
4.2.1	Interface	10
4.2.2	Tunnel SSH	12
4.2.3	Videos	12
4.2.4	Capteurs et Moteurs	12
4.2.5	Affichage 3D	12
4.2.6	Cartographie	12
5	Références	13

Chapitre 1

Introduction

Les fonds marins réunissent aujourd'hui de nombreux secteurs et enjeux, tant professionnels que particuliers. On y retrouve entre autre l'exploration sous-marine, la surveillance et maintenance d'installations professionnelles, ainsi que la cartographie des fonds marins. Tout ces domaines demandent le développement de solutions techniques plus rentables et pratiques qu'une intervention humaine. Notre projet propose ainsi un ROV (Remotely Operated Vehicle) polyvalent et simple d'utilisation à cet effet.

Chapitre 2

Présentation du projet

Un ROV est un robot sous-marin contrôlé à distance et permettant une acquisition d'informations, visuelles ou à partir de capteurs. Notre projet de ROV filoguidé, Nautilus, sera transportable et pilotable à l'aide d'un ordinateur portable. Il permettra d'observer facilement des installations ou des fonds marins à l'aide de caméras. Disposant également de fonctions avancées, le Nautilus sera en mesure de recréer le fond marin d'une zone géographique déterminée par l'utilisateur à partir d'une batterie de photographies prises lors de la phase d'exploration. Les différentes fonctionnalités du Nautilus en font ainsi un outil polyvalent, permettant exploration, maintenance et cartographie des fonds.

Chapitre 3

Cahier des charges

3.1 Analyse Fonctionnelle

3.1.1 Structure

Facilement transportable et peu encombrant.

Contraintes :

- Poids : 2-3kg
- Dimension : 300*200*150mm
- Etanche de norme IP 68

3.1.2 Commandabilité

Commandé à distance par une liaison filaire.

Contraintes :

- Câble : 15m
- Carte intégrée dans le ROV
- FPV (First Person View)
- Piloté au clavier

3.1.3 Milieu d'utilisation

Adapté aux contraintes imposées par son environnement.

Contraintes :

- Eau non salé (moins de 1 g de sels dissous par kilogramme d'eau)
- Eau translucide (transmittance de la lumière entre 75% et 95%)
- Lieu : Piscine, lac
- Ecoulement laminaire
- Courant marin inférieur à 2 noeuds
- Profondeur de 10m (résistant à 2 bars)

3.1.4 Energie

Etre entièrement autonome.

Contraintes :

- Autonomie de 20 minutes

3.1.5 Motorisation

Etre mobile une fois immergé.

Contraintes :

- Propulsion électrique
- Déplacement horizontal (Vitesse maximale de 1m/s)
- Déplacement vertical (Vitesse maximale de 0.5m/s)
- Direction droite/gauche à 360 degrés

3.1.6 Acquisitions

Acquérir et transmettre l'information.

Contraintes :

- Acquisition et retransmission d'un signal vidéo
- Acquisition et stockage de photographies
- Mesure de la pression
- Mesure de la position relative avec signaux GPS

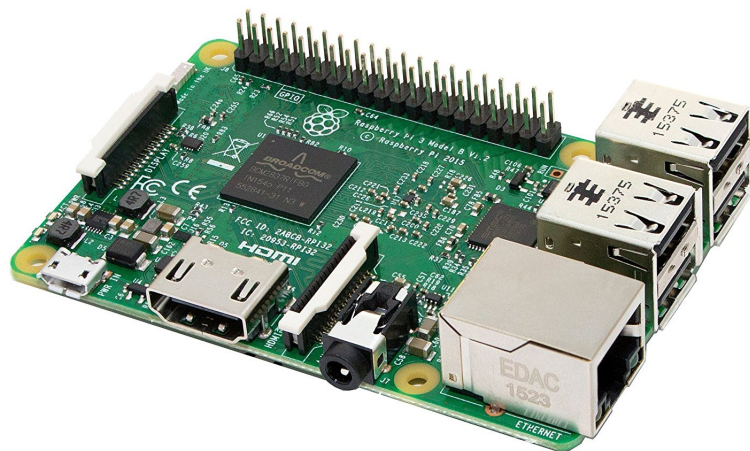
Chapitre 4

Acquisition et Commandabilité

Dans un second temps, nous devons relier les différents éléments de notre ROV sur une carte et ensuite traiter les informations reçus pour pouvoir agir sur les moteurs vus précédemment. Nous avons choisie la Raspberry.

4.1 Raspberry

Une partie de la programmation et des calculs est effectuée sur une Raspberry PI 3 qui supportait tout les types de connections que l'on voulait, en voici la description.

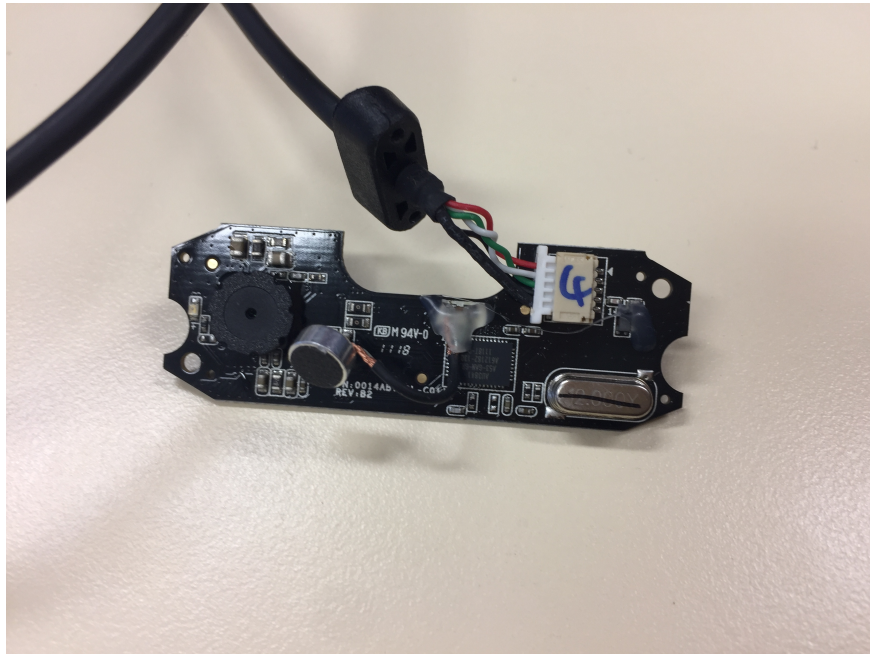


4.1.1 Cameras

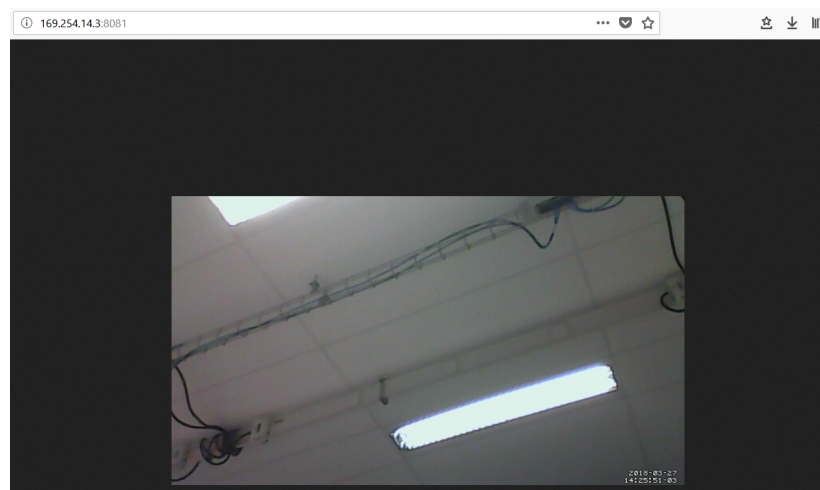
Nous avons 2 caméras qui permettent, l'une la direction (vision frontale) et l'autre la cartographie (vision par dessous). Dans un premier temps, détaillons leur connection entre la Raspberry et le traitement effectué par celle-ci.

Logitech C170

La première est une webcam Logitech C170, que nous avons démonté pour l'assemblage, relié en USB à la Raspberry.



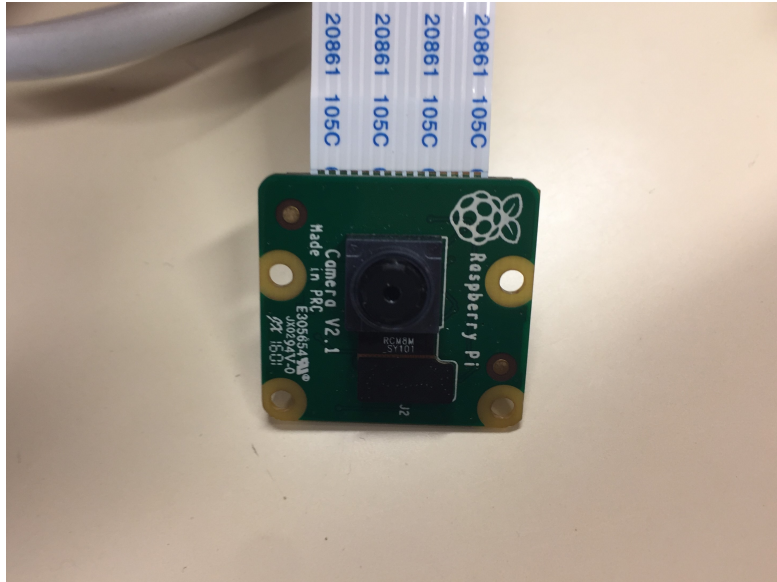
Nous l'avons choisi car le pilote de celle ci est déjà installé nativement sur la Raspberry. Nous utilisons motion qui permet d'envoyer le flux video venant de la camera et de le diffuser en ligne sur notre adresse local (Référence 6). En voici le résultat sur un navigateur :



Cette video sera recuperé par l'interface (expliquer dans le chapitre associé) en 640*360.

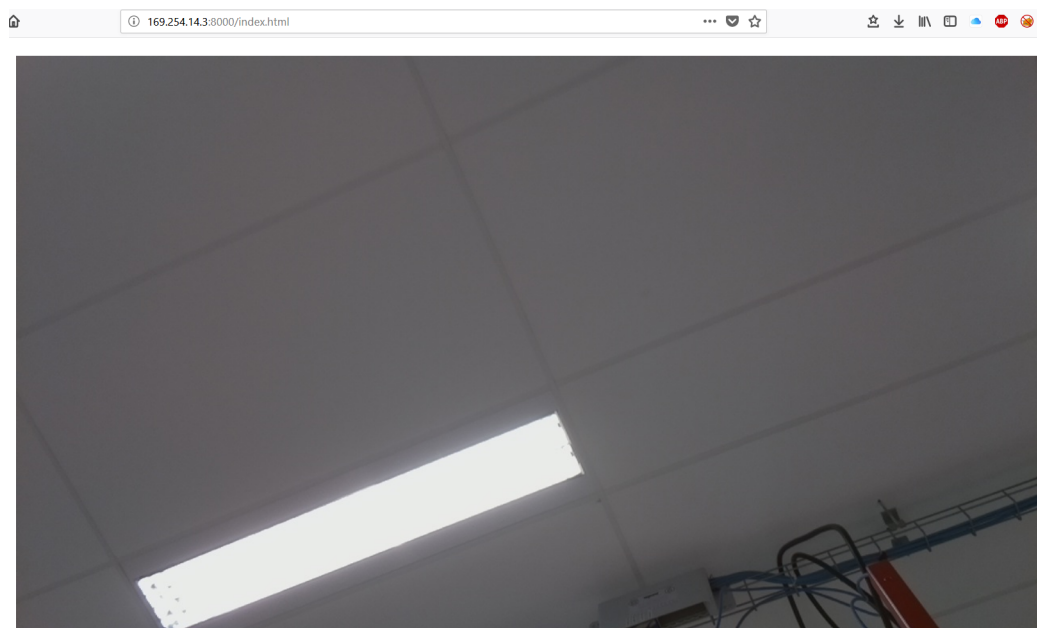
Caméra V2

La deuxième est un module caméra pour raspberry (Référence 7) qui se raccorde directement par une nappe (un bus de type CSI-2).



Elle se paramètre en python avec les librairies données par le constructeur. De même que l'autre caméra, nous renvoyons un flux video en ligne sur notre adresse local (Référence 8) mais cette fois-ci sur un autre port.

Le résultat sur un navigateur :



La video est diffusée en 1920*1080 et affichée par l'interface.

4.1.2 Capteur de Pression/Temperature

4.1.3 Central Inertielle

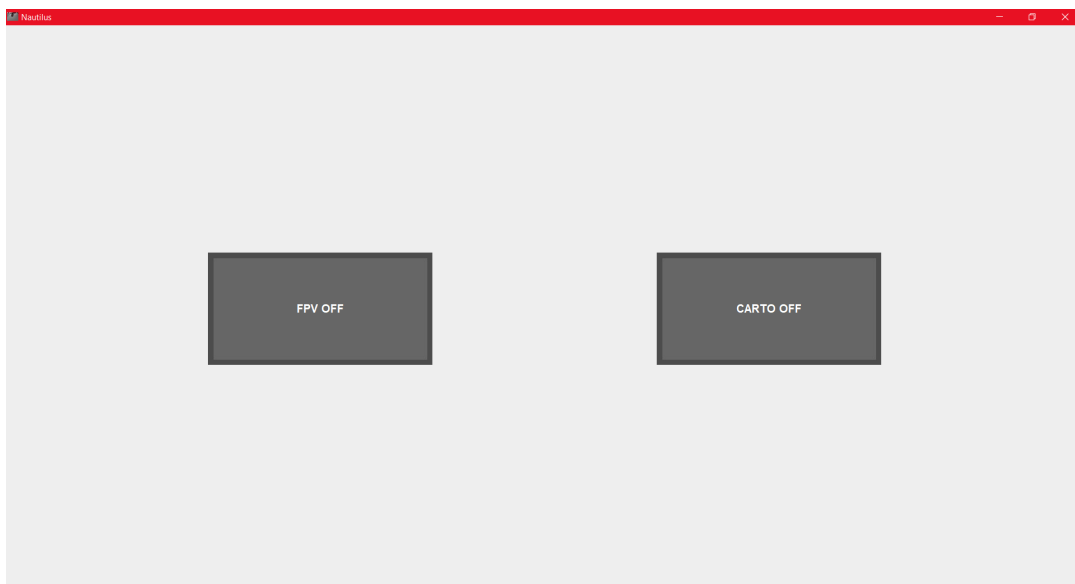
Julien

4.2 PC

Maintenant que nous avons relié tous nos moteurs, caméras et capteurs à la raspberry ainsi qu'un premier traitement des informations, nous allons voir comment nous traitons cela sur le PC.

4.2.1 Interface

En premier lieu parlons de l'interface, celle ci à pris différentes formes au cours du temps, ici nous présenterons que la dernière version. Toute la partie PC a été programmée en JAVA (Disponible ici : Référence 9), l'interface utilise la bibliothèque graphique Swing qui nous permet de gérer l'affichage facilement que ce soit pour la vidéo ou pour les interactions. Lorsque l'application est lancée, nous arrivons donc dans un premier menu :



Le bouton de Droite mène à une partie que nous développerons dans la partie 4.2.6 liée à la Cartographie.

Nous avons donc créé une fenêtre JFrame :

```
1 Interface inter = new Interface("Nautilus",0,0,1920,1080,true);
```

Elle est paramétrée de façon à prendre tous l'écran, ici 1920*1080, les objets se trouvant dans cette fenêtre sont gérés pour se placer en fonction de la taille de l'écran.

Le manager s'appelle GridBagLayout, il nécessite de paramétrer chaque objet. Ce manager crée une grille qui se construit en fonction des paramètres de chaque objet qu'elle contient.

Prenons exemple du premier bouton FPV :

```
1 axPanell1.setMinimumSize(new Dimension(400,210));
2 axPanell1.setMaximumSize(new Dimension(400,210));
3 axPanell1.setPreferredSize(new Dimension(400,210));
4 c.fill = GridBagConstraints.BOTH;
5 c.anchor = GridBagConstraints.CENTER;
6 c.gridx = 0;
7 c.gridy = 0;
8 c.weightx = 0.0;
9 c.weighty = 0.0;
10 c.gridwidth = 1;
11 c.gridheight = 1;
12 c.insets = new Insets(0, 0, 0, 200);
```

Les 3 premières lignes correspondent à la taille du bouton, que nous avons choisis ici de garder fixe.

La ligne 4 n'est pas utile dans ce cas mais permet de correctement redimensionner l'objet lorsque la fenêtre change de taille.

La ligne 5 fixe l'objet au centre de la partie qui lui a été alloué.

La ligne 6 et 7 donne la ligne et la colonne où doit se situer l'objet.

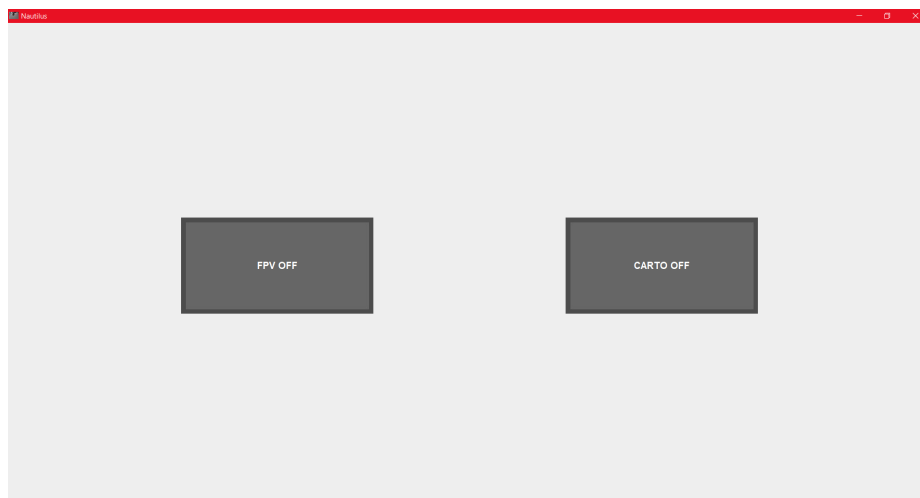
La ligne 8 et 9 définissent des poids en x et y qui sont utilisés lors d'un redimensionnement, cela permet de donner plus de poids à un objet plutôt qu'à un autre. Nous ne l'utilisons pas d'où la valeur 0.

La ligne 10 et 11 permettent de définir combien de ligne et combien de colonne va prendre notre objet.

La ligne 12 insère une marge dans l'ordre suivant (margeSupérieure, margeGauche, margeInférieure, margeDroite).

Chaque objet de notre interface est défini de cette façon.

Ensuite il y a le bouton de Gauche qui lance le système complet. Un nouveau menu remplace le précédent :



Nous avons maintenant la FPV en haut à gauche, le bouton à droite est le même que celui précédemment, les informations des capteurs sont affichées dans le cadre à droite, les commandes envoyées aux moteurs sont en bas et pour finir un affichage 3D avec JAVA3D du ROV en bas à droite. Toutes les informations étant actualisées en temps réel avec la Raspberry. Nous allons voir comment.

4.2.2 Tunnel SSH

4.2.3 Videos

4.2.4 Capteurs et Moteurs

4.2.5 Affichage 3D

4.2.6 Cartographie

Chapitre 5

Références

Motorisation et énergie :

- **Référence 1** : Moteur d'avion électrique brushless ROXXY 315079 chez Conrad (x3)
- **Référence 2** : ESC Suppo Multirotor 20A M20A chez RobotShop (x3)
- **Référence 3** : ESC HobbyKing 30A avec Reverse (x1)
- **Référence 4** : Radiocommande Graupner MX-20 (disponible à l'ENSEA)
- **Référence 5** : Batterie d'accumulateurs (NiMh) 7.2 V 3000 mAh Conrad (x1)

Acquisition et Commandabilité :

- **Référence 6** : Adresse Camera Frontal
- **Référence 7** : Module Camera V2 chez Mouser Electronics
- **Référence 8** : Adresse Camera du Dessous
- **Référence 9** : Code JAVA de l'interface