



Rapport de projet de 2ème année

Année universitaire 2025-2026

Semestre 3

Conception d'un robot poisson pour la surveillance maritime et la recherche en robotique biomimétique

Etudiant :

BIECHE Matys Adéas

Encadrant :

M.LEBARS Fabrice

Ensta Campus de Brest

Engagement de non plagiat

Je soussigné, BIECHE Matys Adéas,

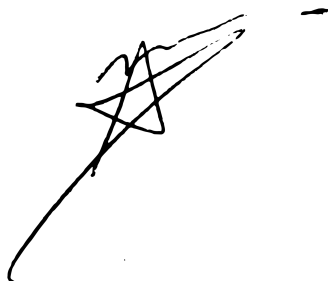
Déclare avoir pris connaissance de la charte des examens et notamment du paragraphe spécifique au plagiat.

Je suis pleinement conscient(e) que la copie intégrale sans citation ni référence de documents ou d'une partie de document publiés sous quelques formes que ce soit (ouvrages, publications, rapports d'étudiant, internet etc...) est un plagiat et constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.

En conséquence, je m'engage à citer toutes les sources que j'ai utilisées pour produire et écrire ce document.

Fait le 23/12/2025

Signature

A handwritten signature in black ink, consisting of a series of loops and a long horizontal stroke extending to the right.

Résumé

Dans le cadre du projet de deuxième année, nous avons travaillé sur la conception et la mise en œuvre d'un robot poisson biomimétique destiné à la surveillance maritime. À partir de l'analyse du besoin, de la compréhension des contraintes mécaniques, électroniques et hydrodynamiques, ainsi que de la définition des fonctions principales du système, chaque étape du développement a été pensée pour aboutir à un prototype fonctionnel, robuste et évolutif.

L'objectif du projet est de proposer une plateforme expérimentale fiable et modulaire, capable d'assurer des déplacements de base (avancer, plonger, remonter) dans un milieu aquatique contrôlé, tout en posant les fondations pour de futures applications en surveillance environnementale et en recherche robotique sous-marine biomimétique.

Ce rapport a pour vocation de présenter la démarche de conception, les choix de composants, ainsi que les choix techniques, les tests expérimentaux à réaliser, mais aussi les perspectives d'amélioration et d'évolution du système pour le semestre 4.

Ce rapport sera donc l'occasion de mettre en lumière les compétences acquises, mais aussi les découvertes et recherches réalisées.

Sommaire

1	Introduction	4
1.1	Contexte & Problématique	4
1.2	Objectif	5
2	Etat de l'Art	6
2.1	Robots sous-marins conventionnels (ROV / AUV)	6
2.2	Robots poissons biomimétiques : principes généraux	6
2.3	Propulsion et architectures mécaniques	7
2.4	Commande, perception et missions de surveillance	7
2.5	Plateformes biomimétiques existantes et solutions commerciales	8
2.6	Positionnement de notre projet	9
3	Cahier des charges	10
4	Robot Poisson Biomimétique	14
4.1	Repository	14
4.2	Mécanique	14
4.2.1	Tail	15
4.2.2	Body	20
4.2.3	Head	21
4.2.4	Assemblage et tests d'étanchéité	22
4.3	Système embarqué	24
4.3.1	Electronique & Capteurs	24
4.4	Informatique	26
4.4.1	ESP32 : Bas Niveau	26
4.4.2	Communication	28
4.4.3	Raspberry Pi : Haut niveau	30
5	Conclusion	30

1 Introduction

Dans le cadre des projets de 2ème année, nous présentons dans ce compte rendu les éléments de conception de notre robot poisson, ainsi que nos recherches sur le sujet de la robotique biomimétique type poisson.

1.1 Contexte & Problématique

La surveillance des milieux marins, qu'il s'agisse du suivi de la biodiversité, du contrôle de la qualité de l'eau, de l'inspection des structures offshore, de la gestion des fermes aquacoles, de l'entretien des éoliennes en mer ou encore de la détection de pollutions, constitue aujourd'hui un enjeu majeur pour la protection de l'environnement et la gestion durable des ressources marines.

Les méthodes conventionnelles, reposant sur l'utilisation de navires, de bouées instrumentées, ou encore de véhicules sous-marins téléopérés (ROV) et autonomes classiques (AUV), présentent néanmoins plusieurs limites. Elles sont souvent coûteuses, génèrent des nuisances sonores susceptibles de perturber la faune marine, offrent une autonomie limitée et rencontrent des difficultés pour atteindre certaines zones ou assurer une immersion prolongée dans des environnements complexes où la visibilité est réduite.

Face à ces contraintes, les robots biomimétiques, imitant le comportement et la morphologie des poissons, apparaissent comme une alternative prometteuse. Grâce à leur design inspiré du vivant, ils peuvent évoluer de manière plus discrète, s'intégrer naturellement aux écosystèmes aquatiques, et dans certains cas, améliorer l'efficacité énergétique de leurs déplacements.

1.2 Objectif

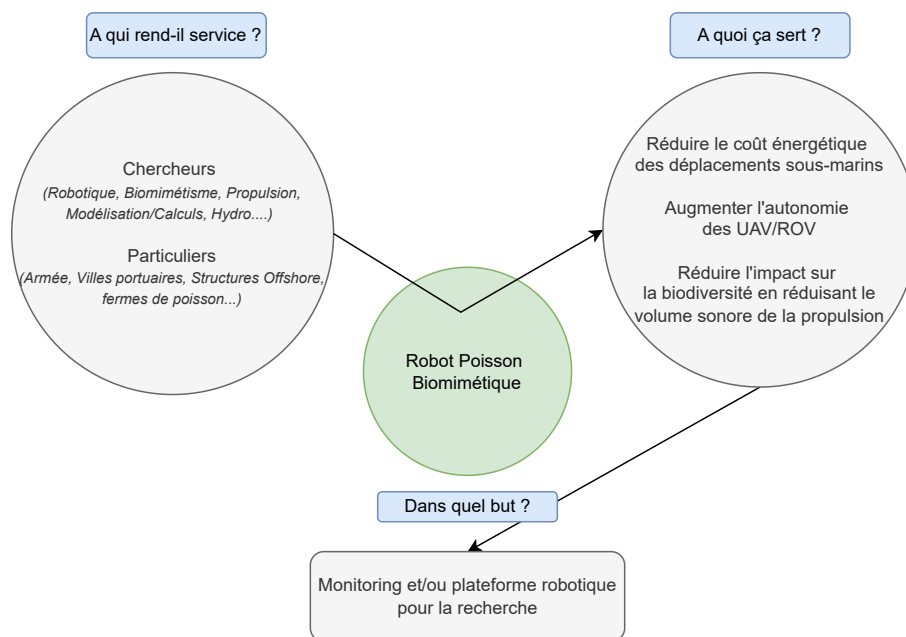


Figure 1.1 *Bête à corne du projet*

Notre objectif est donc de concevoir un prototype de robot poisson biomimétique Open Source, capable de réaliser des déplacements minimaux dans un milieu aquatique, afin de poser les bases d'applications futures en surveillance maritime biomimétique.

Nous devons pour cela partir de zéro et suivre des étapes clés de développement :

- **Concevoir et assembler la partie matérielle du robot poisson :**
 - Réaliser le choix et l'intégration des composants principaux (moteurs, batterie, microcontrôleur(s)...)
 - Assurer l'étanchéité et la robustesse minimale pour des tests en bassin.
- **Mettre en place la partie commande et contrôle :**
 - Choisir et prendre en main la carte de contrôle (microcontrôleur ou Raspberry Pi) pour piloter le ou les actionneurs.
 - Intégrer les capteurs de base nécessaires (capteur de pression et température).
 - Développer les algorithmes de contrôle qui permettront au robot de : Avancer, tourner, plonger et remonter.
- **Définir un mode de fonctionnement minimal :**
 - Programmer des séquences automatiques simples type machine à états (ex. avancer 5 secondes, plonger à une profondeur cible, remonter).
 - Vérifier le bon fonctionnement mécanique (propulsion, étanchéité, flottabilité).

- Essais en bassin pour valider la capacité du robot à avancer, plonger et remonter de façon contrôlée.
- Ajustements rapides pour corriger la stabilité et le comportement du robot.

2 Etat de l'Art

L'étude de l'état de l'art vise à situer notre projet de robot poisson biomimétique par rapport aux solutions de robotique sous-marine existantes, tant du point de vue des architectures mécaniques que des stratégies de propulsion, de commande et des cas d'usage en surveillance environnementale.

2.1 Robots sous-marins conventionnels (ROV / AUV)

Historiquement, la surveillance et l'inspection des milieux marins reposent sur deux grandes familles de systèmes : les véhicules sous-marins téléopérés (ROV) et les véhicules autonomes sous-marins (AUV). Les ROV sont reliés à un navire support par un ombilical, disposent d'une forte capacité d'emport et d'action (bras manipulateurs, capteurs lourds) et sont largement utilisés pour l'inspection d'ouvrages offshore, l'intervention sur pipelines ou l'assistance en grande profondeur. Les AUV, plus autonomes, sont privilégiés pour des missions de cartographie, de prospection géophysique ou de suivi de paramètres océanographiques sur de grandes distances.[4]

Ces plateformes présentent toutefois des limitations structurelles pour des missions de surveillance fine de l'environnement ou de la biodiversité : empreinte acoustique et hydrodynamique importante, signatures visuelles marquées, manœuvrabilité limitée à basse vitesse, coût d'acquisition et d'exploitation élevé. Elles sont peu adaptées à des opérations prolongées dans des zones restreintes, proches d'habitats sensibles ou au voisinage d'organismes vivants que l'on souhaite perturber le moins possible. Ces contraintes ouvrent un espace pour des approches biomimétiques, plus discrètes et potentiellement plus sobres énergétiquement.[4]

2.2 Robots poissons biomimétiques : principes généraux

Les robots poissons biomimétiques s'inspirent directement de la morphologie et de la cinématique de nage des poissons. La littérature distingue classiquement deux grands modes de propulsion bio-inspirés : la propulsion *Body and/or Caudal Fin* (BCF), où la poussée est principalement générée par l'ondulation du corps et/ou de la nageoire caudale, et la propulsion *Median and/or Paired Fins* (MPF), qui s'appuie sur le battement de nageoires médianes et paires pour assurer la propulsion et le contrôle d'attitude.[9] Ces approches permettent d'obtenir des trajectoires plus souples, une excellente manœuvrabilité à basse vitesse, ainsi qu'une signature hydrodynamique et acoustique plus proche du vivant.[9, 5]

Les revues récentes sur les robots poissons soulignent plusieurs atouts majeurs de ces systèmes : meilleure efficacité propulsive à vitesse modérée, capacité à effectuer des virages serrés, meilleure intégration à l'écosystème et réduction de la perturbation de la faune marine par rapport aux véhicules à hélices classiques. Les poissons robots sont aujourd'hui étudiés comme plateformes de référence pour la nage silencieuse, la navigation dans des environnements encombrés (herbiers, récifs, structures portuaires) et la coopération en essaim.[5, 2]

2.3 Propulsion et architectures mécaniques

Sur le plan mécanique, on retrouve trois grandes familles d'architectures de robots poissons :

- **Architectures rigides articulées** : le corps du robot est segmenté (2 à 4 segments principaux) reliés par des articulations actionnées par des servomoteurs ou des moteurs à courant continu. L'ondulation est obtenue par des lois de commande de type sinusoïdal ou quasi-sinusoïdal appliquées aux articulations, ce qui simplifie la modélisation cinématique et la réalisation mécanique.[6]
- **Architectures souples (soft robotics)** : le corps ou la nageoire caudale sont réalisés en matériaux polymères souples (silicone) actionnés par des actionneurs distribués (câbles, chambres pneumatiques, actionneurs diélectriques, etc.). Cette approche permet de reproduire plus fidèlement la flexibilité du poisson et d'améliorer l'efficacité propulsive, au prix d'une complexité accrue de fabrication et de modélisation. Des plateformes open source comme OpenFish illustrent cette tendance, en combinant segments rigides et nageoire caudale souple pour optimiser la vitesse et le rendement propulsif.[1, 7]
- **Architectures hybrides** : combinaison d'un corps rigide pour l'intégration des batteries, de l'électronique et des capteurs, avec une queue ou des nageoires souples ou semi-souples pour la propulsion. Ce compromis offre un bon compromis entre intégration de charge utile, robustesse et performances hydrodynamiques.

Les travaux récents sur les mécanismes de propulsion caudale (revues de mini-mécanismes à bielles-manivelles, mécanismes à glissière double, etc.) montrent une recherche active d'architectures compactes, efficaces et faciles à étanchéifier, adaptées à une intégration dans des coques de petite taille.[6]

2.4 Commande, perception et missions de surveillance

Sur le plan de la commande, la majorité des robots poissons biomimétiques reposent encore sur des stratégies relativement classiques : lois de commande en amplitude/fréquence de l'ondulation caudale, PID pour le contrôle de profondeur ou d'assiette, et régulateurs

plus avancés (commande prédictive, backstepping, LQR) pour la stabilisation de trajectoire. Les travaux les plus récents commencent à intégrer des approches d'apprentissage par renforcement pour optimiser conjointement la forme de la nage et la politique de commande à partir de simulations hydrodynamiques en environnement virtuel, avant validation sur des prototypes physiques.[4, 5]

Pour la perception, les capteurs embarqués se rapprochent progressivement de ceux des AUV classiques : capteurs de pression pour la profondeur, IMU (centrale inertielle) pour la stabilisation, caméras pour la vision sous-marine et parfois sonars légers ou capteurs inspirés de la ligne latérale des poissons (capteurs de pression distribués).[4]). Dans un contexte de surveillance environnementale, les robots poissons sont utilisés ou envisagés pour :

- la mesure de paramètres physico-chimiques (température, salinité, turbidité, pH, oxygène dissous) en zones côtières, ports ou fermes aquacoles ;
- la détection de pollutions (microplastiques, hydrocarbures, rejets chimiques) via des capteurs dédiés ;
- l'observation discrète de la faune marine, notamment pour l'étude des comportements, de la reproduction ou du suivi d'espèces sensibles, en tirant parti du faible niveau de perturbation généré par ces robots ;
- la cartographie locale de structures (digues, coques de navires, fermes aquacoles) en environnement peu profond.[8, 5]

Plusieurs prototypes récents montrent que ces plateformes peuvent être utilisées comme vecteurs de capteurs pour la surveillance d'écosystèmes ou pour la collecte d'ADN environnemental, en s'approchant au plus près des habitats sans induire des fuites massives d'organismes.[8]

2.5 Plateformes biomimétiques existantes et solutions commerciales

Sur le plan applicatif, l'offre se répartit aujourd'hui entre :

- **Plateformes de recherche académiques** : prototypes conçus pour étudier des aspects spécifiques (hydrodynamique de nage, contrôle multi-robots, interaction avec la faune). On peut citer des projets de poissons robots souples (SOFi au MIT, projets EPFL) ou les robots développés au sein de laboratoires italiens ou asiatiques pour explorer l'intégration à l'environnement et la cohabitation avec les poissons sauvages.[3]
- **Plateformes open source** : quelques projets se positionnent explicitement comme supports open source pour la recherche et l'enseignement (OpenFish, plateformes de test modulaires de type « soft robotic fish »), avec mise à disposition des fichiers mécaniques, des schémas électroniques et du code de commande, afin de faciliter la reproductibilité et l'extension des travaux.[1, 7]

- **Produits commerciaux et démonstrateurs industriels** : il existe désormais des poissons robots commercialisés pour l’exploration, la démonstration technologique ou l’éducation, combinant design biomimétique et pilotage sans fil. Certains modèles mettent en avant la nage silencieuse, la furtivité et la capacité de déploiement en essaim pour des missions de surveillance ou d’inspection de bassins.[3]

On observe également l’émergence d’autres systèmes biomimétiques marins (robots méduses, calmars, pieuvres) visant des objectifs similaires de furtivité, de manœuvrabilité et d’intégration dans l’écosystème, ce qui confirme la tendance de fond en direction de la robotique sous-marine bio-inspirée.[11]

2.6 Positionnement de notre projet

Au regard de cet état de l’art, notre projet de robot poisson biomimétique se positionne à l’interface entre :

- les ROV/AUV classiques, dont nous reprenons certaines briques technologiques (architecture de contrôle, capteurs standards, méthodologie de tests en bassin) ;
- les robots poissons de recherche, dont nous adoptons les principes de nage ondulatoire et la logique de plateforme expérimentale pour la robotique biomimétique ;
- les projets open source, en visant explicitement la mise à disposition publique des fichiers mécaniques, du code et de la documentation, afin de permettre la réutilisation et l’extension par d’autres étudiants ou chercheurs.

La contribution principale de notre travail n’est pas de proposer une nouvelle architecture de propulsion radicalement innovante, mais de démontrer qu’un prototype de robot poisson, conçu avec des moyens accessibles à un binôme d’élèves ingénieurs, peut constituer une plateforme expérimentale robuste, modulaire et reproductible pour des applications futures de surveillance maritime et de recherche en robotique biomimétique. Cette approche met l’accent sur la traçabilité des choix, la réutilisabilité des briques matérielles et logicielles, et l’intégration cohérente des fonctions de base (propulsion, contrôle de profondeur, acquisition de données) dans une architecture globale orientée mission.

3 Cahier des charges

De nombreux acteurs ont eu leur importance sur le développement de notre robot. Ces derniers ont défini les objectifs et les contraintes auxquelles il devrait se soumettre au regard des missions qu'il doit pouvoir effectuer.

Nous résumons ces contraintes liées aux interacteurs sous forme du diagramme suivant :

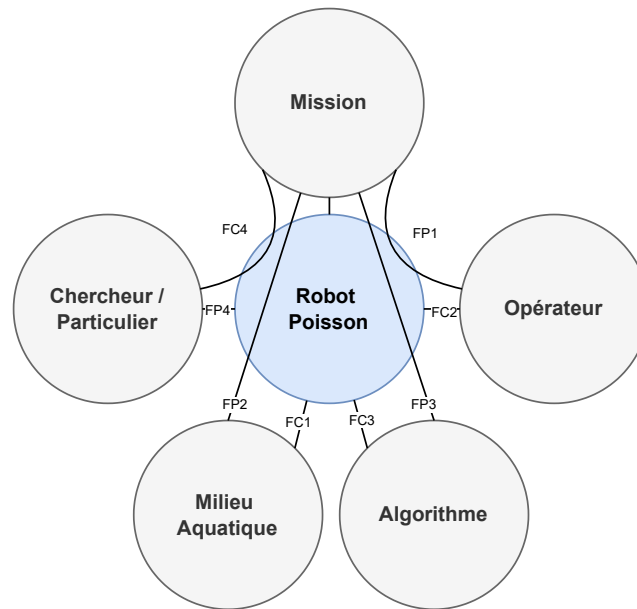


Figure 3.1 *Diagramme Pieuvre de notre système*

Après l'étude de ces acteurs, nous avons défini des fonctions principales (FP) et contraintes (FC). Les fonctions principales doivent satisfaire l'objectif à atteindre tandis que les contraintes permettent d'assurer la sécurité de l'outil, de l'opérateur dans le cadre défini par les exigences imposées :

Liaisons	Type de Fonction	Description
Opérateur → Robot Poisson → Mission	FP1	Lancer une mission programmable que le robot exécute de manière autonome
Mission (capteurs/caméra) → Robot → Milieu	FP2	Percevoir et enregistrer les données utiles à la mission
Algorithme (commande) → Robot → Mission	FP3	Assurer la commande et la stabilisation autonome du robot (avancer, plonger, remonter)
Robot → Chercheur/Particulier	FP4	Publier en open source le code, les schémas, les données et la documentation
Milieu aquatique → Robot	FC1	Résistance au milieu aquatique : étanchéité et tenue en pression
Opérateur → Robot	FC2	Contrôle et prise en main simples
Algorithme → Robot	FC3	Fiabilité et stabilité du système pour un fonctionnement sécurisé
Chercheur/Particulier → Robot	FC4	Modularité matérielle et logicielle pour d'autres missions

Figure 3.2 *Tableau des fonctions minimal*

Le lecteur remarquera que nous avons établis notre analyse au regard de la mission à effectuer. Notre travail vise à répondre à certaines fonctions présentées dans le tableau précédent afin de réaliser la Fonction Principale FP1.

Dans la suite, nous décidons de développer notre travail dans un tableau qui énonce les fonctions traitées, comment elles le sont, de quelle manière, c'est à dire de façon directe en traitant la fonction ou en traitant une autre fonction qui l'impacte puisque 20 % des causes permet de régler 80% des effets et enfin, sur quels critère d'évaluation jugeons-nous notre travail.

Fonction traitée	Comment ?	De quelle manière ?	Critère(s) d'appréciation
FP1	Développement d'une machine à états et d'une interface de lancement de mission	Directement	Lancement fiable d'une mission prédéfinie ; autonomie complète sur la séquence
FP2	Intégration des capteurs de mission (pression, température, caméra) et acquisition synchronisée	Directement	Données cohérentes, fréquence stable, stockage ou transmission correcte
FP3	Implémentation d'algorithmes de commande (PWM, PID) pour stabiliser le mouvement du robot	Directement	Trajectoire stable, vitesse maîtrisée, profondeur maintenue sous perturbations
FP4	Mise en place d'un dépôt open source (code, schémas, documentation, données)	Directement	Dépôt public accessible, licence OSI, documentation complète et à jour
FC1	Conception du caisson étanche, choix des joints et essais en immersion	Directement	Absence de fuite après 40 minutes d'immersion, intégrité du système électronique
FC2	Développement d'un mode manuel via une interface PC	Directement	Réactivité aux commandes, stabilité de contrôle
FC3	Optimisation logicielle et gestion des erreurs système	Indirectement	Aucun crash, latence de calcul stable, consommation CPU maîtrisée
FC4	Architecture matérielle et logicielle modulaire (connecteurs, code réutilisable)	Directement	Ajout possible d'un capteur ou module sans modification majeure du système

Figure 3.3 *Tableau des fonctions traitées*

Le tableau précédent met parfaitement en évidence qu'en changeant / améliorant / ajoutant certains aspects il est possible d'impacter la quasi-totalité des fonctions directement ou indirectement.

Ainsi, notre travail consistera donc à :

- Concevoir et tester une enveloppe étanche pour le robot poisson
- Concevoir et tester une architecture électronique minimale afin d'assurer les fonctions principales du robot en sécurité
- Concevoir et tester une architecture informatique minimale qui permettra au robot d'avancer, de tourner et de plonger via une machine à état
- Réaliser une interface utilisateur
- Documenter et commenter les travaux réalisés

4 Robot Poisson Biomimétique

Afin de mener à bien le projet, nous optons pour une conception hardware-in-the-loop. Le choix des composants et de l'architecture du système dans son ensemble seront détaillés par la suite.

4.1 Repository

Tout notre travail est disponible dans un repository Github temporaire via les liens suivants :

CAO : <https://github.com/AdeMBCH/Projet-2A-Fish-CAO.git>

Software : <https://github.com/AdeMBCH/Projet-2A-Fish-Software-Embedded.git>

4.2 Mécanique

La conception mécanique d'un robot-poisson impose une exigence critique : l'étanchéité. Il s'agit de garantir le débattement fluide de la nageoire dorsale tout en assurant un carter rigide et parfaitement étanche, destiné à protéger et accueillir l'électronique embarquée. La mécanique est donc structurée autour de trois sous-ensembles : la propulsion (queue), le corps (enceinte étanche) et la tête (interface capteurs/commande). Nous avons opter pour de l'impression 3D en PET-G pour les pièces mécaniques dans un soucis de simplification de conception.

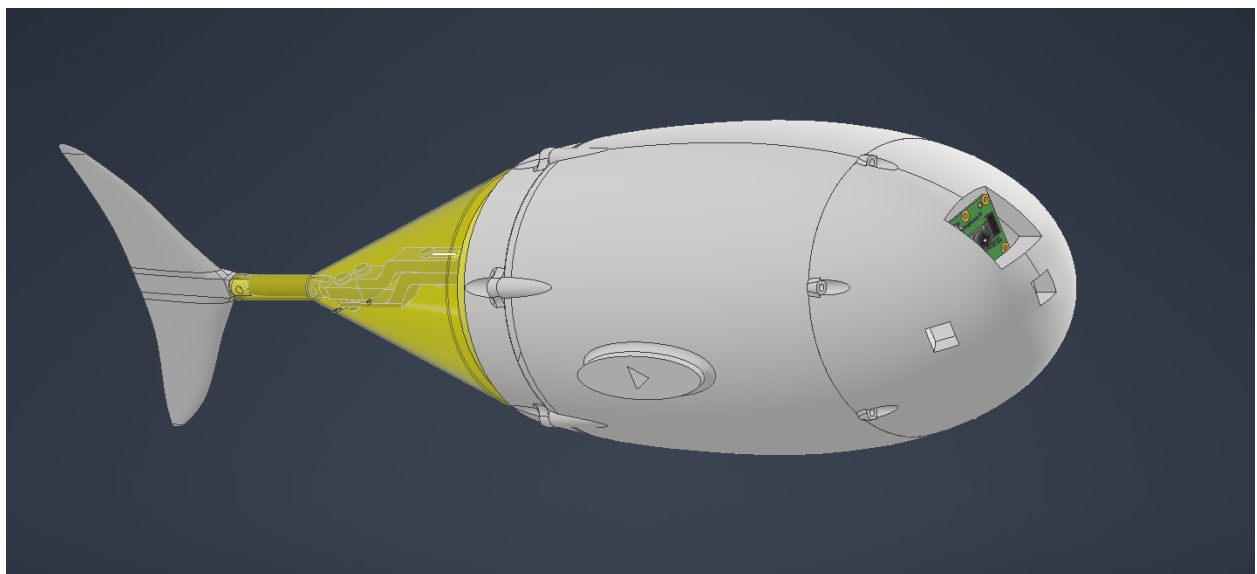


Figure 4.1 *Vue de l'assemblage du robot poisson en CAO sur Inventor*

4.2.1 Tail

Le sous-ensemble « Tail » regroupe le mécanisme de propulsion par oscillation caudale, l'anneau de jointure et l'enveloppe en silicone.

Le mécanisme de propulsion ou « Fish-Tailing » a pour objectif de convertir le mouvement du servomoteur en une déformation contrôlée de la queue afin de générer une poussée efficace. Ce système est repris d'un robot-poisson japonais sur lequel j'ai pu travailler. L'architecture du fish-tailing 3-axes a donc été imaginée par l'ingénieur français Stéphane Weber depuis les travaux de [10]. Nous l'avons repris et retravaillé pour qu'il soit adapté à notre robot.

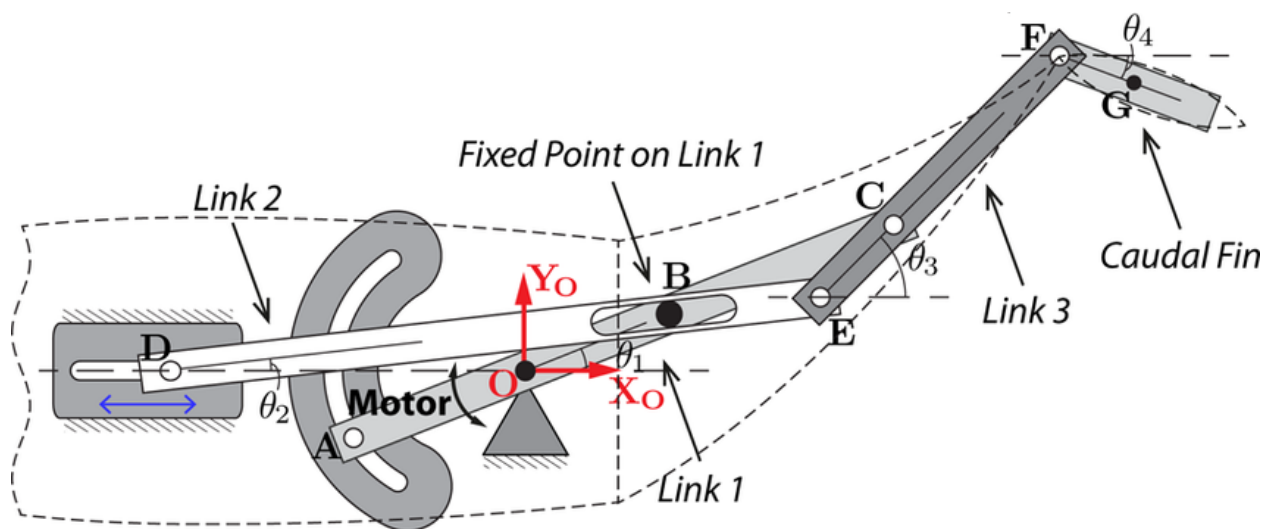
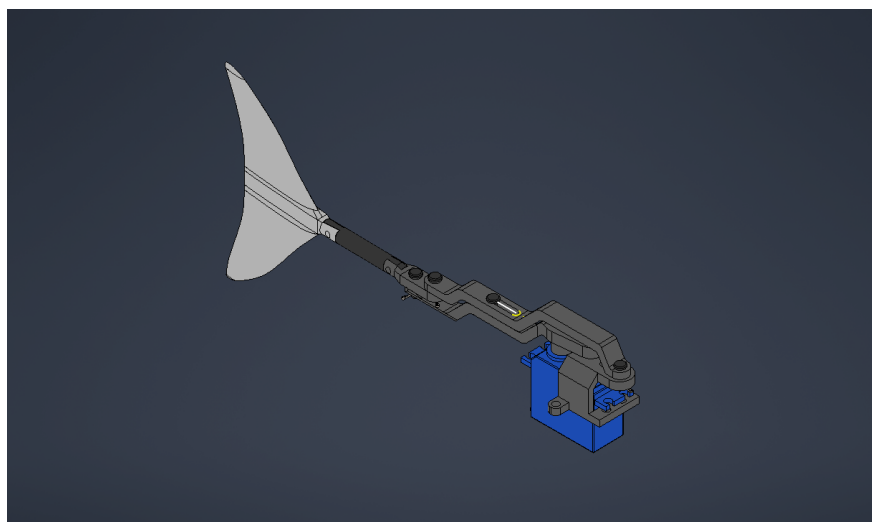


Figure 4.2 Schéma représentatif du système de propulsion



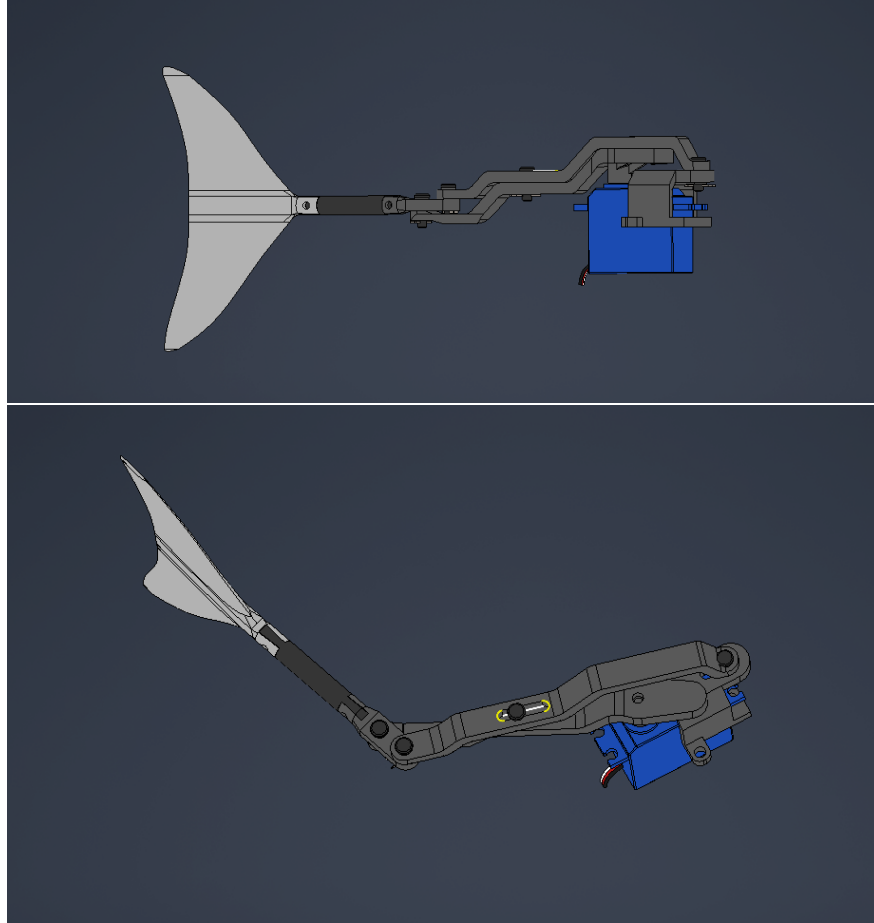


Figure 4.3 *Vue de l'assemblage du mécanisme de propulsion sur Inventor*

Son gros avantage tient dans sa simplicité mécanique. Trois axes et un actionneur suffisent pour reproduire le battement de queue d'un robot poisson.

De plus, tourner peut se faire plus facilement selon l'un des trois modes définis depuis les travaux de [6] :

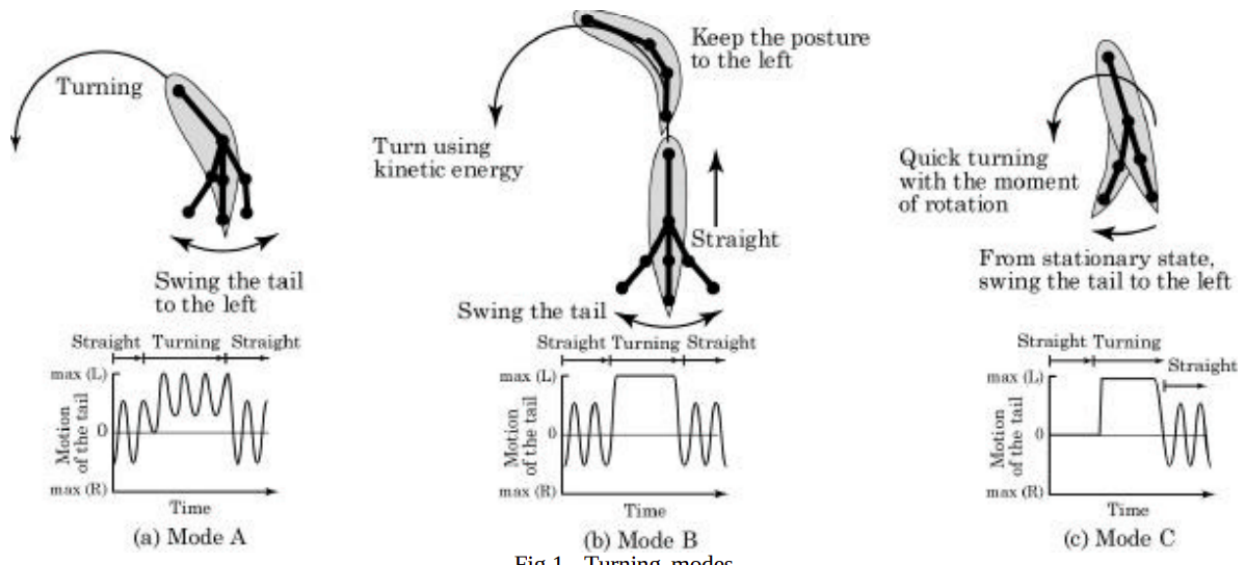


Figure 4.4 Les trois modes pour tourner avec notre mécanisme de propulsion

Nous avons pour l'instant choisi le mode C pour tourner puisqu'il est plus facile à implémenter.

Après impression, le mécanisme est fonctionnel et facilement montable et démontable.

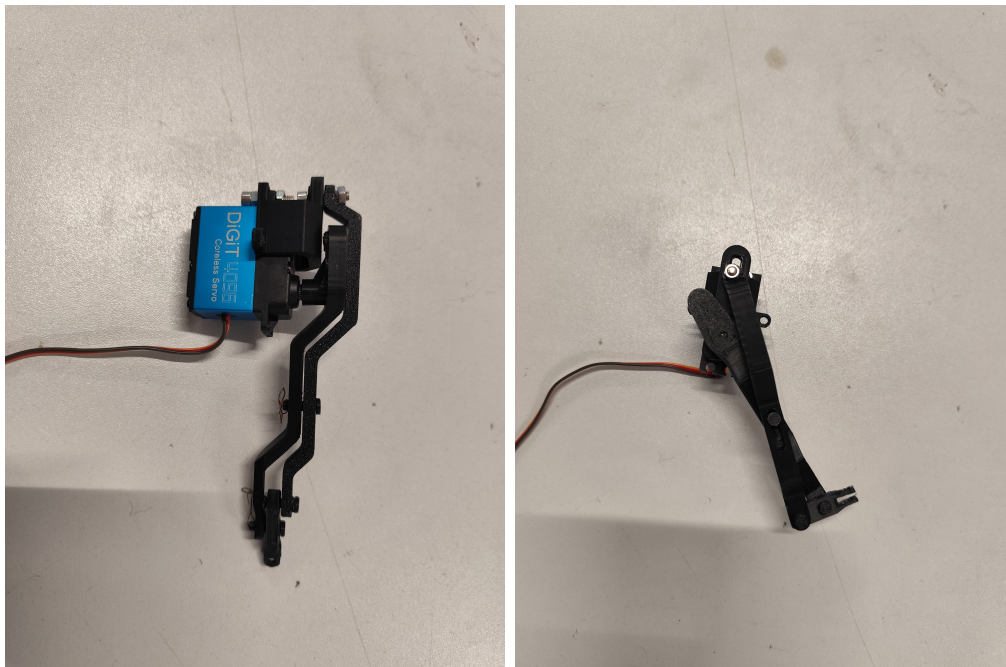


Figure 4.5 Mécanisme de propulsion

En revanche, une pièce s'avère trop fragile pour l'effort qu'elle supporte et pourra être usinée par la suite dans un matériau plus solide.



Figure 4.6 *Assemblage du mécanisme de la queue cassé*

Le système de propulsion est ensuite enveloppé de silicone, servant d'enveloppe étanche, mais aussi et surtout de continuité souple du corps principal.

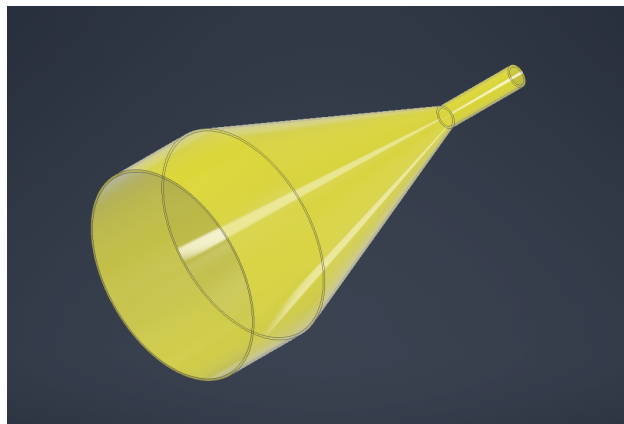


Figure 4.7 *Vue de l'enveloppe en Silicone*

L'enveloppe en silicone est réalisée à l'aide de silicone R PRO 20 de la marque Reschimica. Nous avons donc réalisé un moule en impression 3D afin de pouvoir faire la queue en silicone.



Figure 4.8 *Moule pour le silicone et dosage pour la quantité*

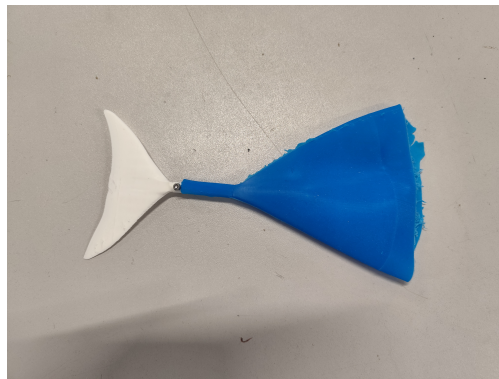


Figure 4.9 *Queue en silicone*

Un anneau muni d'un joint torique permet de rejoindre, d'étancheifier et de consolider la partie Tail avec la partie Body. L'anneau se plaque contre le body et enveloppe le silicone qui sert alors aussi de joint.



Figure 4.10 *Vue de l'anneau de jointure sur Inventor*

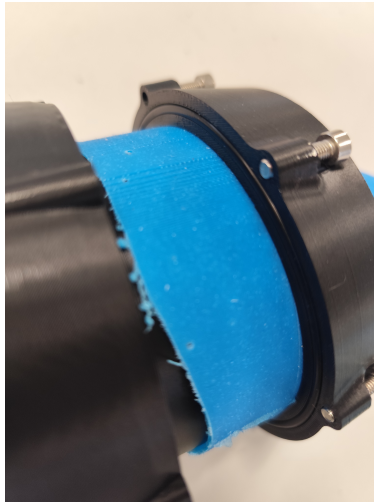


Figure 4.11 *Montage de l'anneau sur le body*

4.2.2 Body

Le « Body » constitue l'élément structurel central du robot et, surtout, l'enceinte étanche destinée à accueillir l'électronique embarquée. Sa géométrie reprend strictement un profil NACA0024 afin d'optimiser l'hydrodynamique et de garantir une nage stable et efficiente.

À l'arrière, le corps intègre le support de l'actionneur de fish-tailing ainsi que le cylindre d'appui sur lequel vient se fixer la queue en silicone, assurant une transmission mécanique propre et reproductible du mouvement. Sur les flancs, une embase dédiée aux nageoires pectorales a été conçue sous forme de plateforme modulaire : elle permet de monter rapidement différents modules de nageoires (géométries, matériaux, rigidités) afin de caractériser l'impact de ces variations sur la manœuvrabilité et les performances de nage.

Deux trous pourront être percés ou fondus dans le corps afin de faire passer des tuyaux pour gérer le balaste et le capteur de pression.

Enfin, à l'avant, le Body comporte un logement usiné pour un joint torique, destiné à assurer la jonction étanche avec la partie Head par compression.

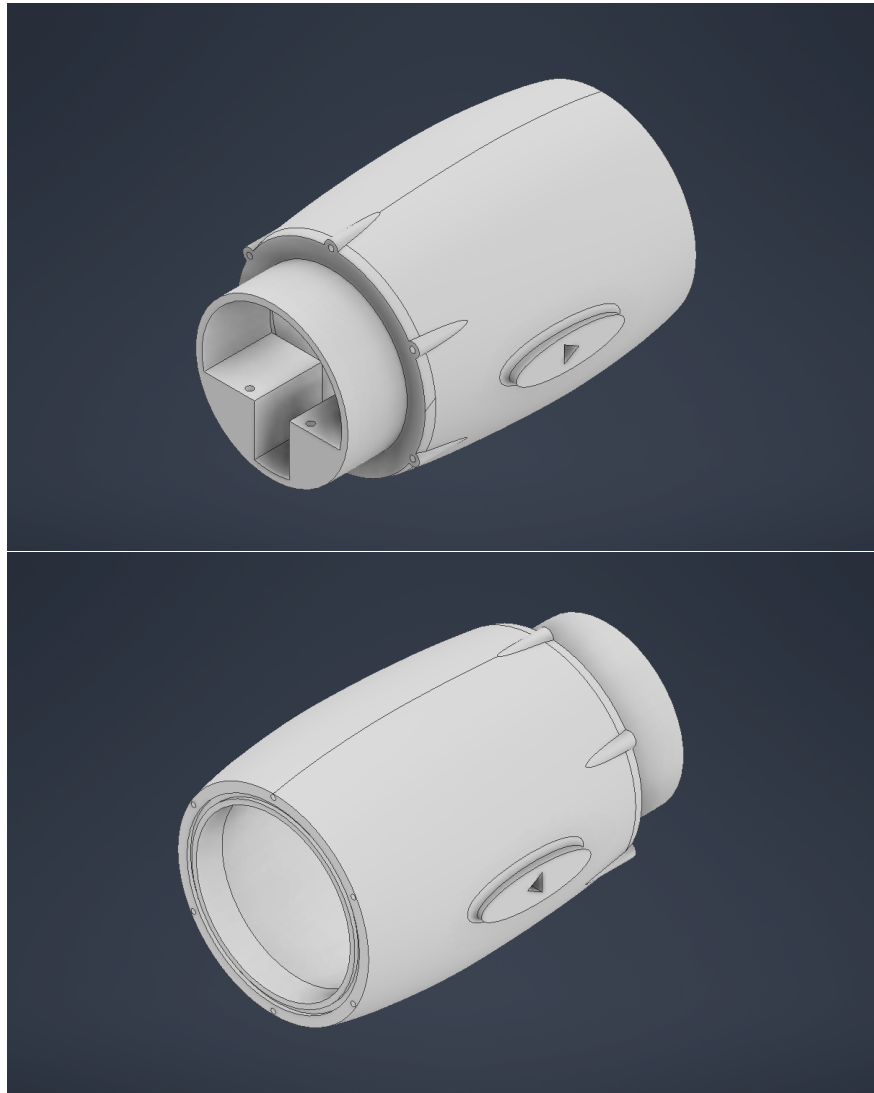


Figure 4.12 *Vues du corps principal sur Inventor*

4.2.3 Head

La « Head » est dédiée à l'intégration des interfaces fonctionnelles situées à l'avant : capteurs (vision, distance), éclairage éventuel et, si nécessaire, éléments de connexion. Elle doit répondre à un double impératif : offrir des zones optiquement fonctionnelles (fenêtres ou ouvertures adaptées aux capteurs) tout en garantissant la continuité de l'étanchéité avec le

Body. Elle constitue également l'ouverture principale du robot, permettant l'accès et l'intégration du matériel électronique.

Initialement, nous avons envisagé l'utilisation du dôme BlueRobotics. Toutefois, cette solution s'est révélée incompatible avec nos capteurs infrarouges, dont le signal ne traverse pas correctement la structure. Nous avons donc conçu une tête spécifique, intégrée dans la continuité du profil NACA0024. Cette pièce comporte quatre ouvertures : trois dédiées aux capteurs infrarouges de distance, et une ouverture centrale réservée à la caméra frontale.

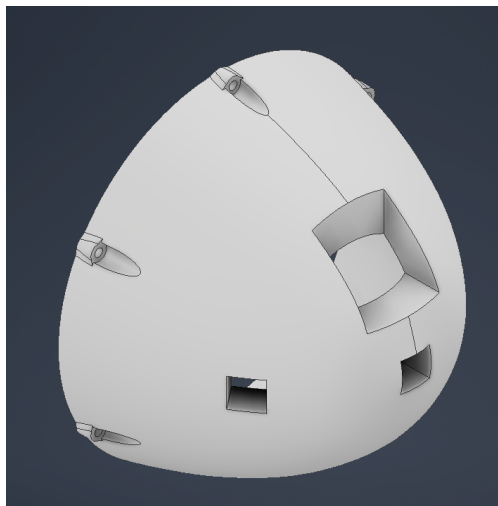


Figure 4.13 *Vue de la tête du robot poisson*

Il faudra couper du plexiglass et utiliser de la colle et du silicone pour former des vitres plates pour la tête.

4.2.4 Assemblage et tests d'étanchéité

Tout au long de la conception et de l'assemblage des sous-ensembles mécaniques, des tests d'étanchéité ont été menés de manière systématique afin de valider chaque interface critique. Les éléments présentés dans ce rapport correspondent aux versions finales ; en pratique, ils résultent d'itérations successives (prototypes, essais en immersion, échecs et améliorations) ayant permis de fiabiliser progressivement la solution retenue.

L'assemblage global s'effectue sans difficulté majeure. Néanmoins, comme indiqué précédemment, une pièce du mécanisme de fish-tailing a été identifiée comme un point de fragilité, en raison des efforts mécaniques importants qu'elle encaisse en fonctionnement. Une amélioration envisageable consiste à usiner cette pièce dans un matériau plus rigide (ou à renforcer sa géométrie) afin d'augmenter la marge de sécurité.

Au final, le robot entièrement assemblé peut être immergé et valide le test d'étanchéité de référence : 40 minutes sous l'eau sans infiltration. (tests réalisés avec le dôme du BlueRov)

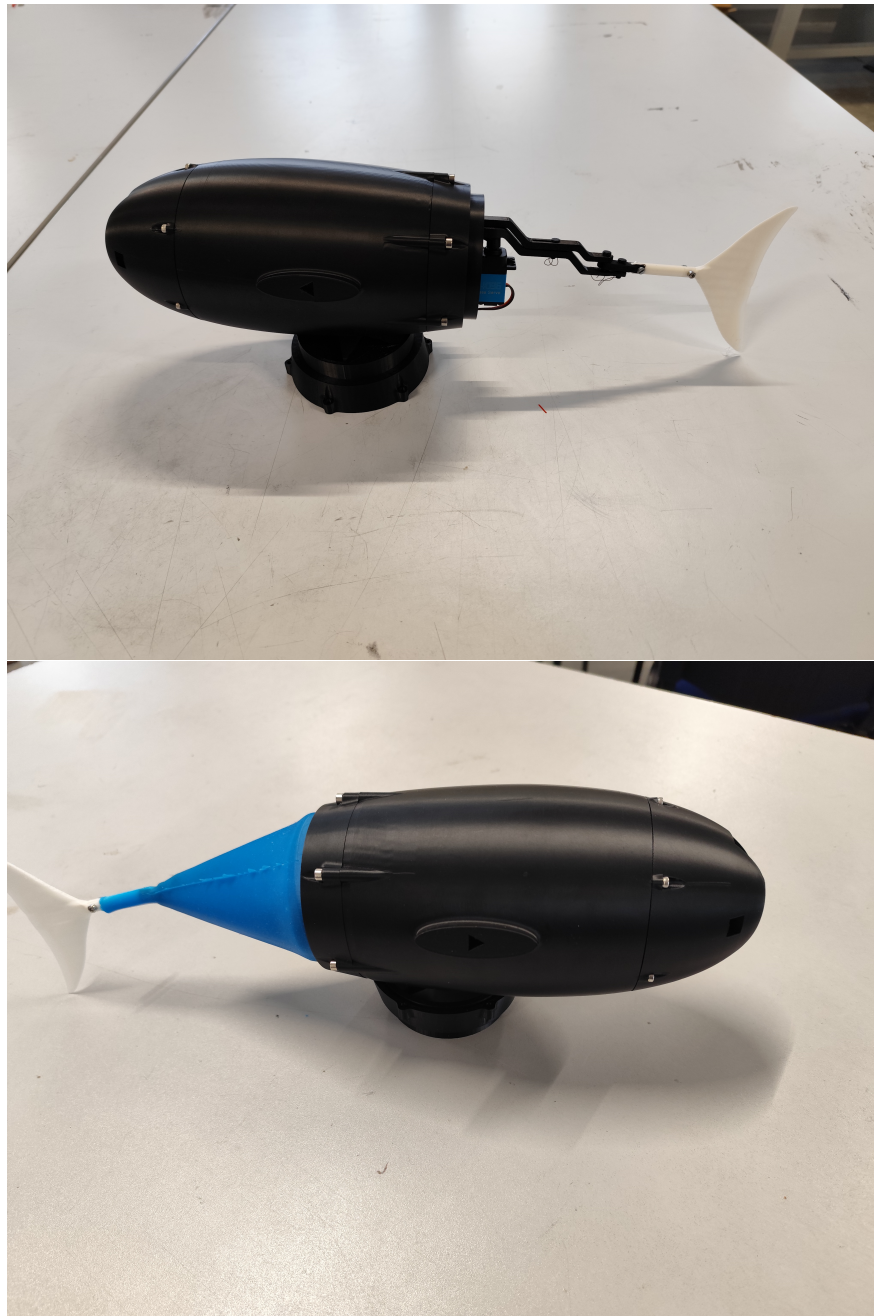


Figure 4.14 *Robot Poisson assemblé*

4.3 Système embarqué

Le robot poisson est conçu suivant une architecture classique de robot, structurée en 4 blocs principaux :

- un bloc **capteurs** pour l'acquisition de l'état et de l'environnement
- un bloc **actionneurs / commande** (servos + carte de pilotage) pour la génération du mouvement
- un bloc **calcul / supervision** pour l'IA et la commande haut niveau
- un bloc **énergie** pour l'alimentation et la distribution de puissance

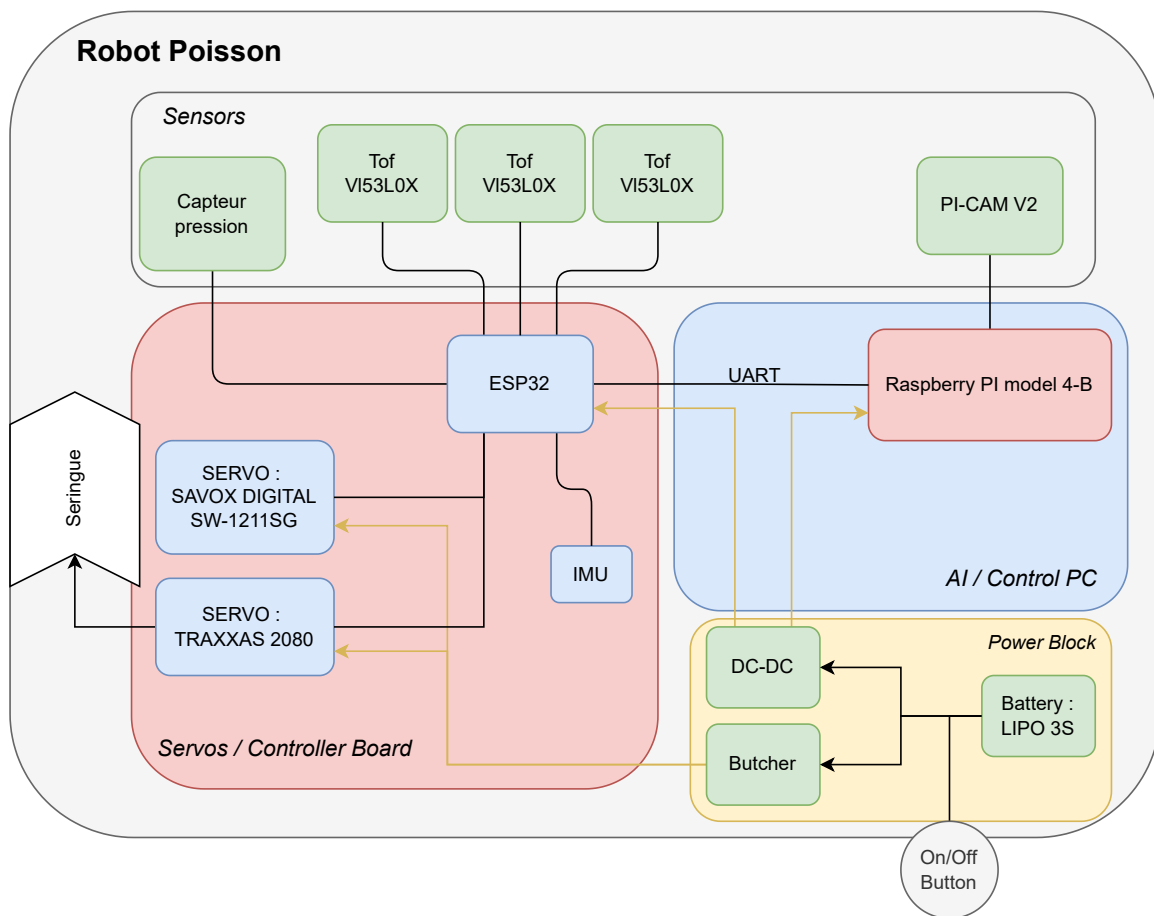


Figure 4.15 Représentation schématique du robot poisson

4.3.1 Electronique & Capteurs

Le robot embarque un ensemble de capteurs complémentaires afin de couvrir à la fois la perception de scène, l'estimation d'état et la mesure de l'environnement proche :

- **Caméra** : une *Pi-CAM V2* connectée à une Raspberry Pi 4 Model B, utilisée pour la perception visuelle (suivi, détection d'éléments d'intérêt, retour opérateur, etc.)
- **Télémétrie courte portée** : trois capteurs **ToF VL53L0X** destinés à la mesure de distance et à l'évitement d'obstacles
- **Profondeur / pression** : un **capteur de pression** pour la mesure de profondeur, exploitable pour la tenue de profondeur
- **Inertiel** : une **IMU** fournissant accélérations et vitesses angulaires, pourra être ajoutée pour l'estimation d'attitude et la stabilisation

L'électronique est organisée autour d'une séparation entre le calcul haut niveau et les commandes bas niveau afin d'isoler les fonctions temps réel des fonctions à forte charge de calcul :

- **Calcul embarqué** : une **Raspberry Pi 4 Model B** assure les traitements de plus haut niveau (supervision, vision via Pi-CAM, logique de mission, etc.)
- **Microcontrôleur temps réel** : une **ESP32** est utilisé pour les tâches déterministes (acquisition capteurs, boucles de commande, génération de consignes actionneurs), avec une liaison **UART** vers la Raspberry Pi.
- **Actionneurs** : deux servomoteurs sont intégrés (**SAVOX Digital SW-1211SG** et **TRAXXAS 2080**) et pilotés via l'ESP32. Une **seringue** est également présente (servant de ballaste).
- **Alimentation** : une batterie **LiPo 3S** alimente le système via un **bloc puissance** comprenant conversion **DC-DC** et distribution. La mise sous tension est réalisée par un **bouton On/Off**.

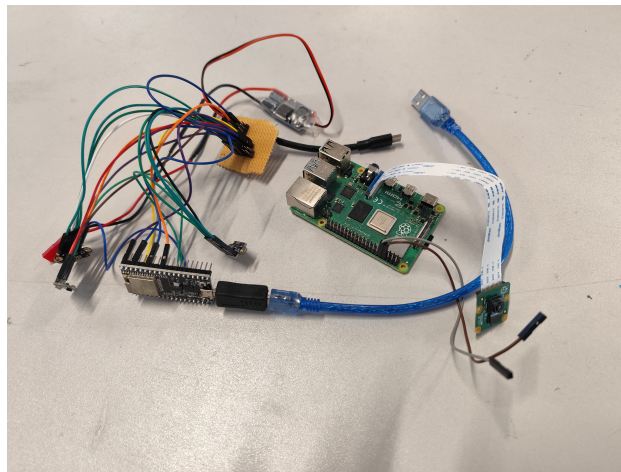


Figure 4.16 *Banc de tests / essais électronique & capteurs*

4.4 Informatique

L'architecture informatique du robot poisson reprend une organisation classique en robotique, structurée en deux niveaux. Le haut niveau, exécuté sur la Raspberry Pi, assure la supervision et la prise de décision globale. Le bas niveau, embarqué sur l'ESP32, garantit l'exécution temps réel des fonctions critiques (commande des actionneurs, lecture capteurs, exécution des comportements locaux). Les deux calculateurs échangent via un protocole de communication UART, décrit ci-près, permettant à la Raspberry Pi d'émettre des commandes vers l'ESP32 et de récupérer des retours d'état.

Le choix d'une architecture temps réel a été retenu afin de maîtriser les latences et d'assurer un comportement déterministe. Le firmware ESP32 est donc basé sur FreeRTOS : le paradigme de programmation s'appuie sur des tâches planifiées plutôt que sur une boucle infinie unique. Cette approche permet de séparer proprement les responsabilités et d'exécuter de façon concurrente plusieurs traitements, par exemple l'évaluation de la machine à états (FSM) et le contrôle d'autres fonctions (comme la profondeur), sans dépendre d'une logique monolithique ni complexifier excessivement la gestion par interruptions.

Enfin, l'ESP32 disposant de deux coeurs, l'ordonnancement FreeRTOS peut être exploité en multi-cœur : certaines tâches peuvent être affiliée à un cœur spécifique tandis que d'autres s'exécutent sur le second. On obtient ainsi un parallélisme matériel effectif en complément de la concurrence logique offerte par FreeRTOS, ce qui améliore l'isolation des charges (par exemple séparer une tâche de contrôle critique d'une tâche de communication) et renforce la stabilité temporelle du système.

4.4.1 ESP32 : Bas Niveau

Le logiciel embarqué orchestre quatre fonctions temps-réel :

- (i) communication UART avec un hôte (supervision/commande),
- (ii) acquisition multi-capteurs ToF (perception obstacles),
- (iii) navigation via machine à états finis (fonctionnement minimal),
- (iv) actionnement des servos (propulsion, virage, profondeur...) par génération PWM.

Le code est structuré en briques quasi-indépendantes (orienté objet) en C :

- **Communication série**
 - `uart_port.c` : driver UART (ESP-IDF) + file d'événements `evt_q`
 - `ring_buffer.c` : buffer circulaire RX découplage ISR/traitement
 - `frame_codec.c` : décodage/encodage de trames + CRC
 - `uart_link.c` : pipeline RX (UART→ringbuffer→decode) + TX (queue→UART)
 - `msg_router.c` : dispatch applicatif par `func` (PING/TEXT/ACTION, etc.)
- **Actionnement (servo)**
 - `servo_hal.c` : abstraction PWM via LEDC (ESP-IDF)

- `servo_tail.c` : modèle de consigne (sinusoïde) + conversion degrés $\rightarrow \mu s$
- `servo_ctrl.c` : contrôle périodique (task) : calcule impulsion et envoie au HAL
- **Perception (ToF)**
 - `tof.c` : init I2C + séquençement XSHUT + lecture VL53L0X + stockage du dernier échantillon
- **Décision & mouvement**
 - `fish_motion.c` : stratégies de mouvement (forward/turn/stop) en réglant le profil servo
 - `fish_fsm.c` : machine à états (IDLE/FORWARD/TURN_LEFT/TURN_RIGHT) pilotée
- **Intégration**
 - `main.c` : init des modules + création des tâches FreeRTOS et attribution des coeurs

Flux fonctionnels (dataflow)

1. **Acquisition** : `tof_task` lit périodiquement 3 capteurs (gauche/centre/droite) et publie le dernier échantillon par capteur (protégé par mutex).
2. **Décision** : `fish_ctrl_task` construit un masque d'obstacles à partir des distances ToF, applique la table de transitions et déclenche l'action d'état (forward/turn/stop).
3. **Commande moteur** : les actions appellent `fish_motion` qui configure le profil sinusoïdal du servo (fréquence/amplitude/centre).
4. **Actionnement** : `servo_task` génère la PWM en continu (périodique), convertit consigne (degrés) en largeur d'impulsion (μs) et l'écrit via LEDC.
5. **Communication (optionnelle)** : le pipeline UART reçoit, decode et route les commandes (et renvoie des réponses) sans bloquer les tâches de contrôle.

Tâches FreeRTOS et contraintes temps-réel Le design découple les latences (UART, I2C) du contrôle servo :

- `servo_task` : boucle périodique stricte via `vTaskDelayUntil` (5 ms) pour une commande stable.
- `tof_task` : boucle périodique (50 ms) suffisante pour l'évitement réactif.
- `ul_uart_rx_task` / `ul_frame_decode_task` / `ul_uart_tx_task` : pipeline asynchrone, avec notification (`xTaskNotifyGive`) côté RX et budget CPU côté decodeur pour éviter la monopolisation CPU.

Paramétrage

- I2C ToF : SDA GPIO21, SCL GPIO22, 400 kHz ; XSHUT GPIO25/26/27 ; adresses forcées 0x30/0x31/0x32.
- Servo : GPIO18 (queue), GPIO19 (ballaste)
- UART : UART2, TX GPIO17, RX GPIO16, 115200 bauds.

4.4.2 Communication

UART : couches et découplage

- **Driver** (`uart_port.c`) : initialisation ESP-IDF (`uart_driver_install`, `uart_param_config`, `uart_set_pin`) et lecture/écriture `uart_read_bytes`/`uart_write_bytes`.
- **Tampon RX** (`ring_buffer.c`) : structure `head/tail` (compteurs monotones). Accès $O(1)$ par masque `idx & (size-1)` (*pré-requis : taille puissance de 2, typiquement 2048*). Ce tampon absorbe les rafales UART sans bloquer le traitement applicatif.
- **Pipeline** (`uart_link.c`) :
 - `ul_uart_rx_task` : consomme la `evt_q` du driver UART, lit les octets disponibles et les écrit dans le ring-buffer ; en cas d'overflow (`UART_FIFO_OVF`/`UART_BUFFER_FULL`) purge l'entrée (`uart_flush_input`) puis notifie le décodeur.
 - `ul_frame_decode_task` : attend une notification, draine le ring-buffer et pousse les octets dans `fc_push_byte`. Lorsqu'une trame est complète et valide, elle est publiée en file `q_frames_rx`. Un budget (`BUDGET=512`) + `taskYIELD()` évite les sections longues.
 - `ul_uart_tx_task` : lit `q_frames_tx`, encode (`fc_encode`) puis écrit sur l'UART.

Protocole de communication Implémenté dans `frame_codec.c`.

Format logique :

Start Of Frame (SOF)	Command	Payload Length	Payload	Checksum
0x41	2 octets	2 octets	n octets	1 octet

- `SOF = 0x4A`.
- `FUNC` : identifiant de commande 16 bits (routage applicatif).
- `LEN` : longueur payload 16 bits, contrôlée vs `payload_buf_sz` (sinon reset).
- `CRC` : `CRC8 XOR` (accumulation XOR de `FUNC`/`LEN`/`PAYLOAD`). En cas de mismatch, reset de l'automate.

Côté émission, `fc_encode` reconstruit le buffer et calcule le CRC à la volée.

Routage applicatif des messages `msg_router.c` implémente une boucle bloquante (`mr_run`) :

- consomme `ul->q_frames_rx`
- dispatche par `m.func` vers des callbacks (`on_ping`, `on_text`, `on_action`)

Cela isole l'I/O du code métier : l'application ne manipule que des messages décodés (`DecodedFrameMsg`) et non des octets.

PWM servo : pile HAL → modèle → tâche

- **HAL** (`servo_hal.c`) : LEDC configuré à 50 Hz ; conversion largeur d'impulsion (μs) en duty 16 bits, puis `ledc_set_duty`/`ledc_update_duty`.

- **Modèle** (`servo_tail.c`) : générateur sinusoïdal

$$\theta(t) = c + a \cdot \sin(2\pi ft)$$

puis conversion degrés $\rightarrow$ μ s via interpolation linéaire entre `us_min/us_max` (avec saturation).

- **Contrôle** (`servo_ctrl.c`) :
 - `sc_set_profile(f,a,c)` configure `freq/amp/center` du modèle.
 - `servo_task` exécute périodiquement : `st_step_compute_pulse_us` $\rightarrow$ `servo_hal_write_pulse`. La périodicité est verrouillée par `vTaskDelayUntil(&last, pdMS_TO_TICKS(5))`.

Perception ToF (VL53L0X) : init multi-capteurs et publication d'état Dans `tof.c` :

- **I2C** : init bus (400 kHz) via ESP-IDF.
- **Adressage multi-capteurs** : tous les VL53L0X démarrent à l'adresse 0x29, donc le code séquence les broches XSHUT :
 1. mettre tous XSHUT à LOW (capteurs éteints)
 2. activer un capteur à la fois, vérifier l'ID modèle, puis écrire un nouvel I2C_SLAVE_DEVICE_ADDRESS (0x30/0x31/0x32)
- **Acquisition** : `tof_read_blocking` lance la mesure (`SYSRANGE_START`), attend un temps de mesure, lit le bloc résultat (`RESULT_RANGE_STATUS`) et extrait la distance (mm). Les derniers échantillons sont stockés par capteur avec mutex.
- **Accès consommateur** : `tof_get_last_sample(index,&sample)` fournit une lecture thread-safe au contrôleur.

Décision : machine à états + anti-oscillation Dans `fish_fsm.c` :

- Construction d'un masque obstacles (LEFT/FRONT/RIGHT) en comparant les distances ToF à un seuil (`obstacle_threshold_mm`, typiquement 100 mm).
- Table de transitions statique `g_sm[state][mask]` vers `next_state` et `action`.
- **Anti-oscillation** : pendant un virage, le code impose une durée minimale (`min_turn_ticks` $\approx$ 1 s) avant de réévaluer les obstacles, évitant les basculements gauche/droite à haute fréquence.
- Les actions d'état appellent `fish_motion:fm_forward, fm_turn_left, fm_turn_right, fm_stop`.

Mouvement : profils paramétrables Dans `fish_motion.c`, les paramètres (fréquence, amplitude, centre, offset de virage) sont regroupés dans `FishMotionParams`. Les fonctions de mouvement ne génèrent pas directement la PWM : elles reconfigurent le *profil* du servo via `sc_set_profile` (ce qui maintient une séparation nette entre décision, consigne, et actionnement).

Point d'entrée et intégration `main.c` instancie les buffers (ex. ring-buffer RX 2048, payload 512), initialise les modules (servo, ToF, UART) et crée les tâches `xTaskCreatePinnedToCore`. Certaines parties (UART + FSM + ToF) sont prévues et déjà câblées mais peuvent être activées/désactivées suivant le scénario de test (commentaires dans `app_main`).

4.4.3 Raspberry Pi : Haut niveau

La Raspberry Pi joue le rôle de **superviseur** : elle exécute la logique haut niveau (stratégie, orchestration, monitoring) et pilote le bas niveau (ESP32) via **UART**. Concrètement, côté Raspberry Pi, l'objectif logiciel est de :

- émettre des **commandes** vers l'ESP32 (ex. ping, action, paramètres),
- recevoir des **retours** (télémétrie, ACK/NACK, messages texte, etc.),
- assurer la **robustesse du lien** (timeouts, reconnexion, filtrage des trames invalides),
- permettre l'**affichage**, la configuration et le contrôle via une interface utilisateur.

Les programmes actuels côté Raspberry Pi se structurent autour de deux éléments principaux :

- `uart_bridge.h/.cpp` : brique de communication série et protocole (encodage/décodage).
- `main.cpp` : application de test/supervision minimale (boucle d'exploitation, affichage).

Même si le haut niveau n'est pas finalisé, la base est déjà alignée avec l'architecture finale, toujours orientée objet. Le lien UART est implémenté en C++ en s'appuyant sur les API POSIX. La Raspberry Pi réplique le protocole et la pipeline de communication utilisés côté ESP32 pour garantir l'interopérabilité.

L'implémentation haut niveau de la Raspberry Pi n'est pas encore finalisée. Il restera donc à concevoir toute la logique d'envoi de commande et d'affichage des états reçus depuis l'esp32 pour la finaliser.

5 Conclusion

En conclusion, le travail réalisé durant ce semestre a posé les bases du robot poisson. Ce projet nécessite la prise en compte de nombreux paramètres dont certains que nous n'avions pas initialement anticipés.

Malgré les nombreux freins et difficultés rencontrées, nous avons pu réaliser un gros travail dont nous sommes fiers et dont les leçons à tirer nous serviront dans la suite du projet.

Par la suite, du point de vue hardware, la tête doit être terminée notamment avec les vitres en plexiglass. Pour la partie software, le haut niveau devra être finalisé. Enfin, toute la logique de contrôle en profondeur devra être ajoutée.

Références

- [1] Vikram Ajsmer et al. Openfish : Biomimetic design of a soft robotic fish for high speed locomotion. *Soft Robotics*, 2022.
- [2] S. Ibrahim et al. A review of a biomimicry swimming robot using smart actuators. *International Journal of Advanced Computer Science and Applications*, 12(11) :390–399, 2021.
- [3] R. K. Katzschmann, J. DelPreto, R. MacCurdy, and D. Rus. Exploration of coral reefs with an underactuated soft robotic fish. *Science Robotics*, 3(16) :ear3449, 2018.
- [4] Dong Li et al. A survey on control and perception techniques for robotic fish. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 47(12) :3000–3014, 2017.
- [5] Shuaishuai Ma et al. A review of robotic fish based on smart materials. *Micromachines*, 14(6) :1178, 2023.
- [6] Sayyed Farideddin Masoomi, Stefanie Gutschmidt, XiaoQi Chen, and Mathieu Sellier. The kinematics and dynamics of undulatory motion of a tuna-mimetic robot. *Journal of Intelligent Material Systems and Structures*, 2015.
- [7] HKUST Soft Robotics Lab. Openfish : An open source soft robotic fish, 2022. <https://softrobotics.hkust.edu.hk/2022/10/16/openfish-biomimetic-design-of-a-soft-robotic-fish-for-high-speed-locomotion/>.
- [8] Luke Speller et al. Shoal : A shoal of underwater robots to protect the environment, 2013. <https://projects.research-and-innovation.ec.europa.eu/en/horizon-magazine/shoal-underwater-robots-protect-environment>.
- [9] Michael S. Triantafyllou, George S. Triantafyllou, and Dick K. P. Yue. Hydrodynamics of fishlike swimming. *Annual Review of Fluid Mechanics*, 32 :33–53, 2000.
- [10] University of Canterbury. Bio-inspired aquatic robots, uc-ika series, 2015. <https://www.canterbury.ac.nz/engineering/schools/mechanical/research/gutschmidt/underwater-robotics>.
- [11] W. Xu et al. Lm-jelly : Liquid metal enabled biomimetic robotic jellyfish. *Advanced Functional Materials*, 32(52) :2208384, 2022.