POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Design and Experiments of a Bioinspired Aquatic Snake Robot

Tesi di Laurea Magistrale in
Mechanical Engineering - Ingegneria Meccanica

Author: **Luca Lanzetti, Daniele Mariana**

Student ID: 945157, 946180
Advisor: Giovanni Bianchi
Co-advisors: Simone Cinquemani
Academic Year: 2022-23

Abstract

Aquatic snakes proved to be very good swimmers, capable of excel in maneuverability without lack in other performance, such as speed or energy efficiency. The aim of this project is to design an aquatic snake robot having such features created by using the biomimetic approach. The robot moves in two dimensions and is swimming on a plane slightly below the water surface. The snake robot was designed to be waterproof without the usage of any external cover, in order to have a smooth surface directly interacting with water, so to improve hydrodynamic performances. This thesis describes all the steps adopted to design and build the robot both from a mechanical and electronic prospective. To support the overall design process, the robot behaviour is simulated using a MLS-MPM plug-in code inside Unity. Simulation's results are used to size the motors of the joints, to optimize the external shape of the robot and to evaluate the swimming performances with several combinations of parameters, such as frequency, amplitude and phase shift between joints. The robot was tested and it showed the capability to generate proper thrust, to perform steering maneuvers in the desired direction and to autonomously avoid obstacles. Finally, the experimental results of forward swimming were compared with the simulation's ones, in order to detect an optimal region of suitable parameters that maximize the speed for this particular geometry.

Keywords: Bioimimetics, Bioinspired robotics, Aquatic snake, Snake robot, Modular robot, Unity simulation

Abstract in lingua italiana

I serpenti acquatici si sono dimostrati ottimi nuotatori, capaci di eccellere nella manovrabilità senza sacrificare altre prestazioni, come la velocità o l'efficienza energetica. L'obiettivo di questa tesi è progettare un robot serpente acquatico con tali caratteristiche, utilizzando un approccio biomimetico. Il robot si muove in due dimensioni e nuota su un piano leggermente al di sotto della superficie dell'acqua. Il serpente robot è stato progettato per essere impermeabile senza l'uso di coperture esterne, al fine di avere una superficie liscia in contatto diretto con l'acqua, con l'obiettivo di migliorare le prestazioni idrodinamiche. Questa tesi descrive tutti i passaggi adottati per progettare e costruire il robot sia dal punto di vista meccanico che elettronico. Per supportare l'intero processo di progettazione, il comportamento del robot viene simulato con l'aiuto di un plug-in di Unity che utilizza l'algoritmo MLS-MPM. I risultati della simulazione vengono utilizzati per dimensionare i motori dei giunti, ottimizzare la forma esterna del robot e valutare le prestazioni di nuoto con diverse combinazioni di parametri, come frequenza, ampiezza e sfasamento tra i moduli. Il robot è stato testato e ha dimostrato di essere in grado di generare la spinta adeguata, eseguire manovre di sterzata nella direzione desiderata e di evitare autonomamente gli ostacoli. Infine, i risultati sperimentali del nuoto lungo una traiettoria rettilinea sono stati confrontati con quelli della simulazione, al fine di individuare una regione ottimale di parametri idonei che massimizzino la velocità per questa particolare geometria.

Parole chiave: Bioimimetica, Robotica bioinspirata, Serpente acquatico, Serpente robotico, Robot modulare, Simulazione in Unity

Contents

Introduction

A snake robot is a robotic system engineered to emulate the movement of a living snake. With the usage of a Bio-inspired strategies, taking inspiration from the resilient and steady locomotion of biological snakes, these robots hold promise in addressing the escalating demand for robotic mobility in unfamiliar and demanding environments. As an example, a snake robot would be able to access highly confined spaces as earthquake-damaged buildings, collapsed tunnels or even nuclear contaminated areas where human intervention is dangerous [41].

Among the various demanding habitats, the underwater realm presents a unique set of challenges for traditional robotic platforms. Harsh and dynamic aquatic environments demand specialized designs capable of navigating through confined spaces, negotiating complex terrain, and executing precise maneuvers. Serpentine locomotion of aquatic species represents a promising solution to these challenges if effectively and efficiently applied to robotic systems. Being able to execute these tasks while maintaining the electrical inner core separated from the fluid would allow an aquatic snake robot to perform deep-sea exploration or even conduct inspections of underwater pipelines, oil rigs and shipwrecks, where high pressure, darkness and extreme temperatures arise.

Recent solutions addressed the challenge to emulate the fluid behaviour of the snake's body with the usage of advanced materials [64] or exploiting complex mechanical designs [20]. However, the simplest strategy to reproduce and efficiently control the sinusoidal movement of a snake is to segment it into sub-sequential, equally lengthed joint modules.

Objective of the Thesis

The objective of this thesis is to create an aquatic snake robot able to be extensively tested on the field, in order to evaluate its performances in terms of maximum achievable speed and maneuverability, as well as the capability to autonomous avoid obstacles. The starting point was an existing aquatic snake robot previously built in PoliMi within the frame of a master's thesis [49]. Due to its intrinsic weaknesses, it was not possible to improve it: for this reason, the robot was entirely redesigned and built.

Thesis Organization

This thesis is organized as follows:

Chapter 1 In this chapter, based on the existing literature, the sea snakes anatomy and their swimming gaits are introduced.

Chapter 2 The State of the Art is summarized, distinguishing between terrestrial and amphibious/aquatic snake robots.

Chapter 3 This chapter focus on the first version of the robot, describing its main characteristics and the emerged features faced.

Chapter 4 The solutions to the emerged issues that led to the design and manufacturing of the second version of the robot are exposed.

Chapter 5 This chapter focus on the simulation used to test both the robots, explaining the main theoretical passages behind the algorithm and the strategy used to create it.

Chapter 6 All the different control levels that ensure the aquatic snake robot's correct behaviour are exposed.

Chapter 7 In this chapter, all the experimental tests performed on the field and the associated results are described.

Chapter 8 The conclusion of the thesis and some possible future developments to proceed with the work are discussed.

1 | Physical Principles of Aquatic Snake Locomotion

There are several sea snake species, they belong to the Elapidae family, they have a close resemble to terrestrial snakes, indeed, sea snakes began their evolution from terrestrial ones about 8-17 million years ago and about 60% of their species started speciating about 1.5-7.5 million years ago [55]. Most of the snakes are venomous and completely adapted to the water, only few of them can survive on the land [47]. Sea snakes mostly prefer warm and shallow water, living close to coastal areas. They are widely spread around the world and can be found from the Indian Ocean to the Pacific Ocean [30]. Figure 1.1 shows the worldwide distribution of one of the most investigated sea snakes, the *Pelamis Platurus* also known as the Yellow-Bellied Sea snake [8, 16, 24].

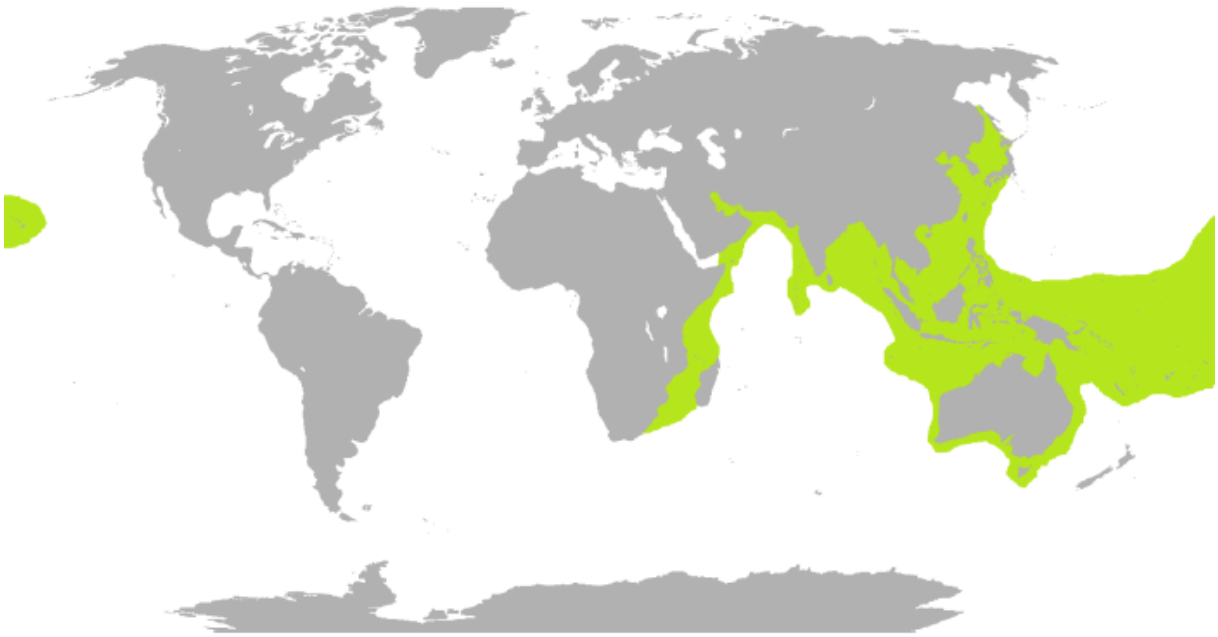


Figure 1.1: Yellow-Bellied Sea Snake distribution around the world [16]

This snake can grow about 45-70 cm, and they can swim as fast as 20 cm/s [24]. However,

most adult sea snake species, in length, grow between 120 and 150 cm and the largest *Hydrophis Spiralis* can grow up to 3m [58].

1.1. Sea Snakes Anatomy

The body of the sea snake has evolved to increase their swimming ability: compared to terrestrial snakes, it has developed a flat tail and became horizontally compressed [14, 44]. This lateral compression had gotten rid of the ventral scales found in terrestrial snakes. The lack of ventral scales on sea snakes makes it difficult for them to survive on land, however, some of them such as Laticauda, were able to retain their ventral scales and have become amphibious: these animals are also known as the Sea Kraits[12]. Scalation among the sea snakes is highly variable since they have developed their own scaling to protect against abrasion. As an example, reef dwelling species have adapted imbricate scales to protect against the sharp corals while some other species have developed scales such as keeled, spiny, smooth, or granular. Pelamis has body scales that are 'peg-like' while those on its tail are juxtaposed hexagonal plates [44].
Although they share the same swimming style, these marine animals are different from the eel: sea snakes have no gills; therefore, they must come to surface often to breathe. Their nostrils have valves consisting of a special spongy tissue to exclude water and the windpipe can be drawn up to where the short nasal passage opens into the roof of the mouth. These animals must partially submerge when breathing. They have long lungs, whose extension is almost the length of whole body and it is believed that the rear portion of this lung is used to balance the buoyancy of the Sea snake [44, 58]. Regarding breathing, some other sea snakes can have cutaneous respiration, in which gas exchange occur across the skin. For example, *Pelamis platurus*, shown in Figure 1.2, obtains about 25% of its oxygen using cutaneous respiration [13].

Figure 1.2: Pelamis platurus [7]

The sea snake has several sensing capabilities, among which, vision, chemo-reception, and hearing.

Due to the extreme flexibility of its body, that allows swimming through narrow spaces and taking sharp turns while in motion, the sea snake swims between coral reefs and caves to catch its preys. The range of the motion of a joint is about 10° to 20° on the horizontal plane and about 2° to 3° on the vertical plane [26]. This unique ability is due to the structure of their skeleton, composed of a number of vertebrae ranging from 130 to more than 400, as shown in Figure 1.3.

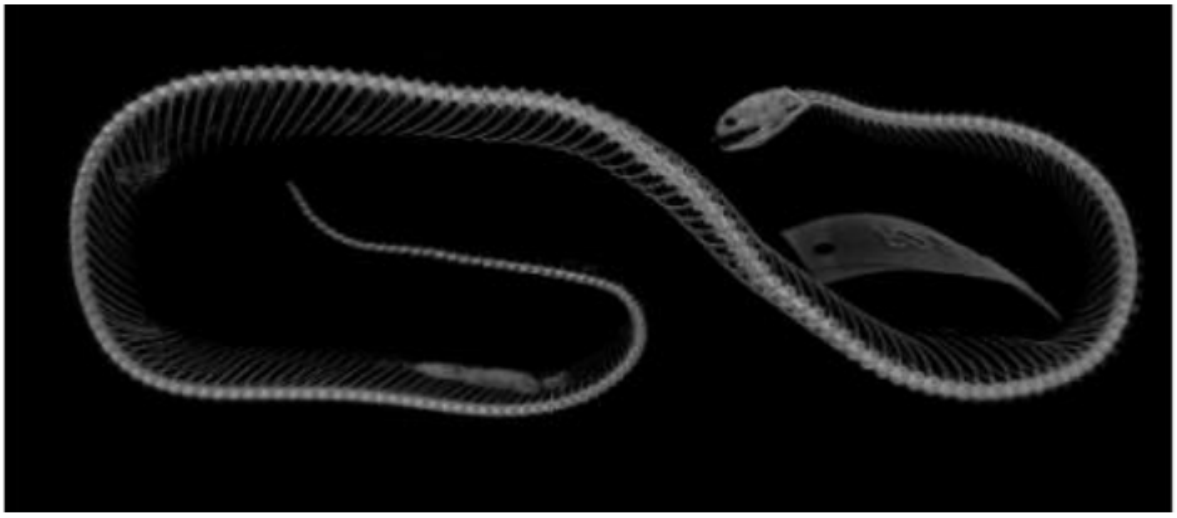


Figure 1.3: Snake skeleton [22]

1.2. Swimming Gait

The swimming pattern of the snake is known as Anguilliform [26], a swimming gait performed by bending the body into backward moving propulsive waves along the body. Sea snakes can swim backward and forward by simply changing the direction of wave propagation. Unlike the fish, the sea snakes use the whole body to generate large amplitude undulations. As reported by Sfakiotakis et al., at least one complete wavelength can be observed on the sea snakes on the motion, and it results in the cancellation of the lateral force which helps to minimize the recoil on the body [54]. The swimming style of the snake change depending on the water depth [24]. When the sea snakes swim, they tend to have their head above the water, their body slightly above the surface and their tail completely immersed as shown in Figure 1.4 [24].

The main swimming pattern of the sea snake observed in planar swimming is known as lateral undulation swimming. It is basically a traveling sine wave along its body as shown in Figure 1.4.

Figure 1.4: Lateral undulation performed by a sea snake [31]

Another marine animal that performs an Anguilliform swimming gait is the eel. Although eels are not in the same taxonomic category of sea snakes, their motion patterns are similar. The main difference between these motions is that the head of eel in eel motion has relatively lower oscillation than the undulatory swimming. Figure 1.5 shows an eel performing its swimming gait. Eel-like motion can be also observed in sea animals like lampreys [56].

Figure 1.5: Eel-like swimming performed by an Eel [51]

Sea snakes have a limited number of swimming gaits, unlike terrestrial ones that have several, such as lateral undulation, concertina, sidewinding and rectilinear.

Anguilliform kinematics is now extensively studied, to highlight the connection with natural swimmers physiology, enhance the realization of efficient bio-inspired underwater vehicles [36]. The work conducted by Khalid et. al explains through numerical simulations the reason that drives natural anguilliform swimmers to employ shorter wavelengths with respect to their bodylength, as emerged from experimental observations conducted on different animal species by [23, 29, 40, 50, 61, 62]. As appeared in the analysis, larger wavelengths enhance the generation of the pressure component of the axial force but, increase the frictional drag over the anguilliform entire length, canceling out any possible advantage [36].

As shown by the work done by Gautreau et al., in Figure 1.6, different snakes show different body movements. Indeed, through the video processing it is possible to extract, directly on the field and with limited animal handling, the different swimming cones and extract all kinematics [20, 21].

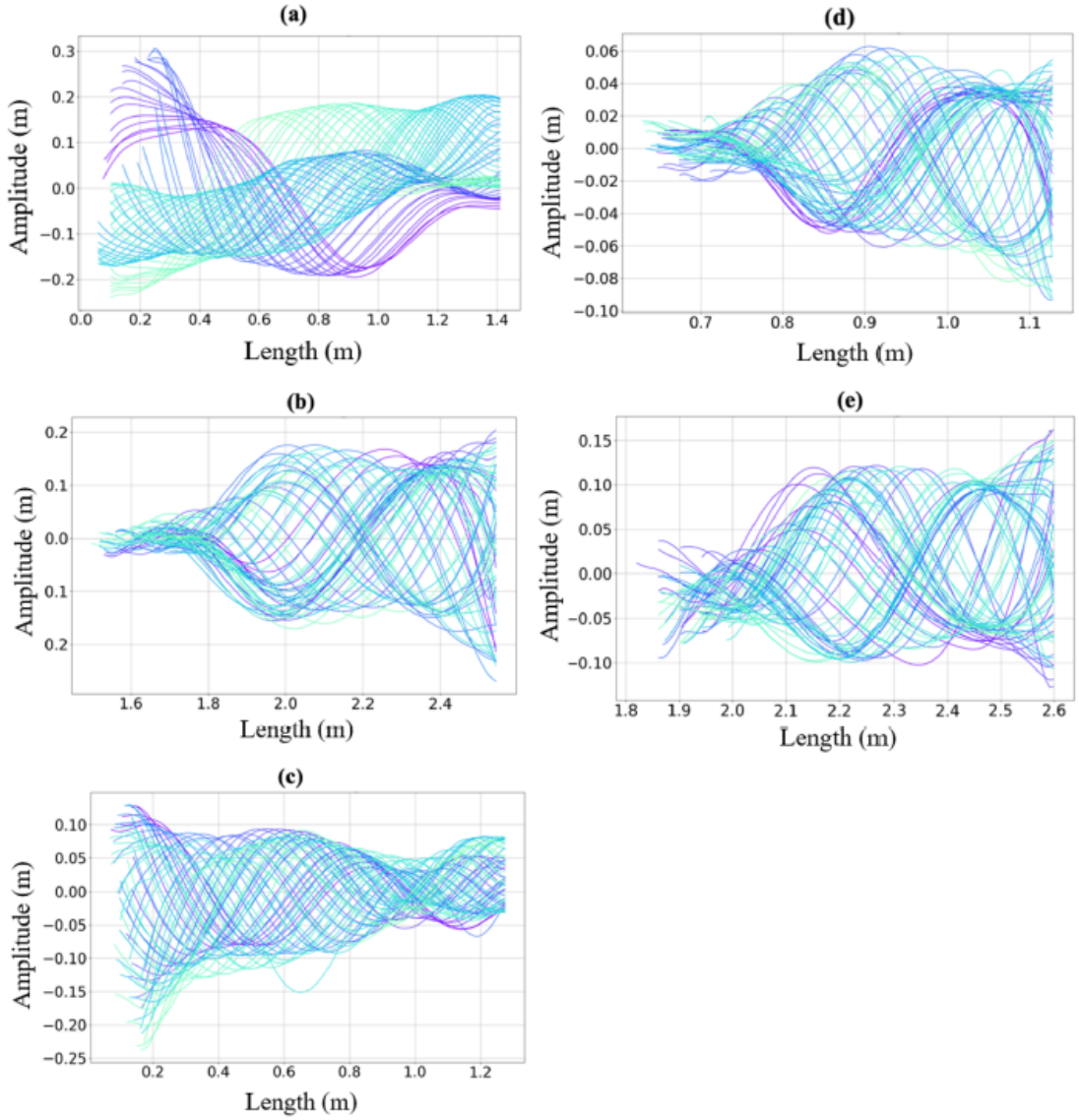


Figure 1.6: Swimming cones extracted from video processing algorithm in [21], **a** Crotalux atrox, **b** Python regius, **c** Hierophis viridiflavus, **d** Vipera aspis, **e** Natrix Helvetica.

This swimming patters data are of fundamental importance to design a novel aquatic snake robot able to mimic the fluid movements of snakes, as done in the work of Gautreau et al.

2 | State of the Art

Numerous studies have been conducted on aquatic snake robots, which can be categorized based on their operational environments: terrestrial, amphibious, and aquatic snake robots. Various actuation techniques have been employed in constructing these robots, including artificial muscle mechanism [46], pneumatic-based [52], and modular-based systems.

Artificial muscle mechanisms were built using different strategies: Giant Magnetostrictive Alloy (GMA), Shape Memory Alloy SMA), electrostatic film, PZT film or Piezoelectric Fiber Composite (PFC). The aim of those studies is to create a flexible structure that has good propulsion characteristics and movement performance that are very similar to real fish-like swimmers. However, building a robot with a flexible structure makes difficult both the design and the control because of the interaction that this structure has with the water. Pneumatic-based systems are very effective for designing robots with high degrees of freedom, the compressibility of air in fact seems to be a suitable feature for building such robots. The design logic consists of a main pipe that transfers pressurized air toward pneumatic actuators that control the compressed air flow through a built-in control valve [52]. However, in this thesis, only modular-based robots will be discussed, due to the choice of designing a robot having such characteristics.

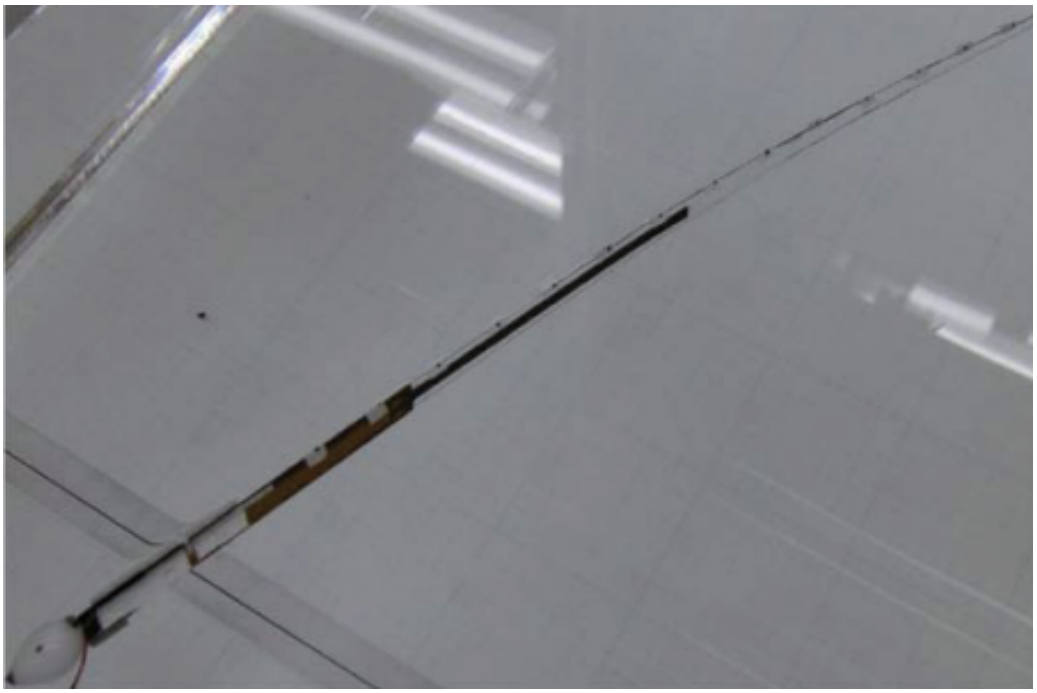


Figure 2.1: Prototype of snake robot built using piezoelectric fiber composite [46]

2.1. Terrestrial Snake Robots

The initial studies in snake robotics concentrated on terrestrial locomotion. The existing works in this domain can be classified into four primary categories based on their interaction with the ground:

1. Passive-wheels: These robots rely solely on internal torque generated through actuated joints for propulsion [17, 48].
2. Wheel-less: While also driven by internal joints, these robots lack wheels, resulting in higher friction compared to passive wheel snake robots [9, 39].
3. Active-wheels: In addition to internal torque, these robots employ wheels to contribute to their motion [42, 68].
4. Elongation: These robots incorporate linear actuators between the modules, supplementing the motion generated by actuated joints [63].

2.1.1. Passive Wheels

The ACM-3, developed by Hirose in 1972, marked the inception of snake robotics. Comprising 20 links and passive wheels, this cable-controlled snake achieved 2D motion with

a weight of 28 kg and a length of approximately 2000 mm. It reached a maximum speed of 400 mm/s [25, 26]. The subsequent ACM-R3, also remotely controlled and equipped with passive wheels, introduced 3D movement. It weighed 12.1 kg and measured about 1755 mm in length [26, 48, 48]. Zhu et al. contributed with a passive-wheel snake robot featuring 4 links, equipped with an onboard power source and wireless computer control [69].

2.1.2. Wheel-less

Bayraktaroglu et al. developed a wheel-less snake robot consisting of 9 modules capable of 2D motion. With a weight of about 900 g and a total length of around 600 mm, it achieved a maximal speed along linear trajectory of about 0.05 m/s [9]. Yim et al. introduced the PolyBot and Polypod, both wheel-less snake robots. Worst and Linnemann worked on a computer-controlled snake-like robot utilizing a CAN bus data connection [65]. Wright et al. designed a snake robot and developed a skin for it, enhancing friction but potentially impeding joint movement and causing heating issues [66].

Wakimoto et al. specialized in developing a snake robot for pipeline traversal. This robot, composed of 13 links, 1187 mm long, and weighing about 1.1 kg, utilized pipe wall friction for movement, achieving a top speed of approximately 48.5 mm/s. The addition of silicone rubber on the outer cover increased adhesion. Notably, the diameter of the pipe influenced the snake's velocity, with larger diameters resulting in decreased speed. Wakimoto et al. planned to construct a waterproof model for enhanced versatility [39].

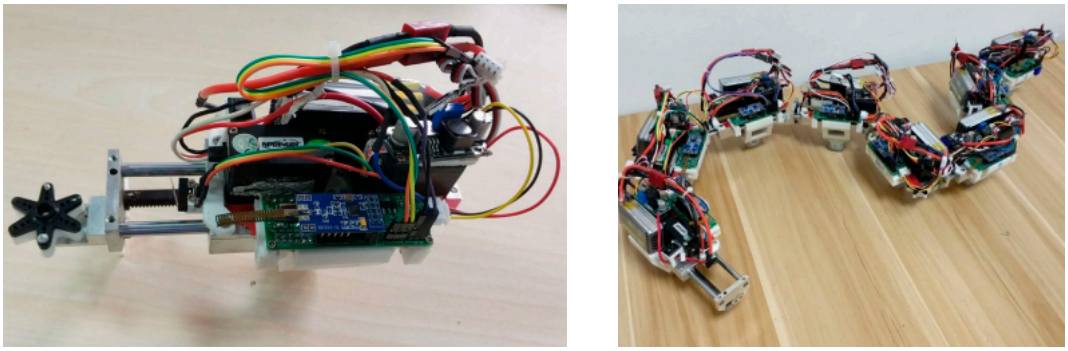
2.1.3. Active Wheels

The Mamba snake robot, developed by Liljebäck et al., could switch between passive and active wheel modes [42]. Hirose and team introduced the ACM-R4, a snake robot capable of driving its wheels, reducing the necessity for many links to generate internal torque. With a length of 1100 mm and a weight of about 9.5 kg, it exemplified this innovation [68]. Genbu-3, another creation by Hirose and team, featured independently driven large wheels and had its own power source, albeit controlled via cable. It measured 1700 mm in length and weighed around 35 kg [37].

2.1.4. Elongation

Wang et al. developed a snake robot capable of elongating between modules, enhancing rectilinear motion to speeds of up to 20 mm/s [63]. Each joint is designed to be 2-

DOF, capable of both rotating and translating, leading to two different gaits, which are serpentine and rectilinear locomotion.



(a) Single module. (b) Assembled robot.

Figure 2.2: Elongating snake-like robot for high environmental adaptability

2.2. Amphibious and Aquatic Snake Robots

Amphibious and aquatic snake robots are of particular interest for this project, both of them are capable of performing aquatic locomotion, with the difference that amphibious one can be considered either aquatic or terrestrial depending on the task. Aquatic snake robots are primarily classified based on the propulsion method, three main categories can be identified:

- Active propeller: These snake robots employ an actuated mechanism to generate additional thrust alongside serpentine movement [11, 59].
- Passive propeller: These robots incorporate non-actuated mechanisms, such as fins, which passively generate thrust alongside serpentine movement [15, 35].
- Propeller less: These snake robots rely only on serpentine movement for propulsion, lacking additional thrust mechanisms [17, 18, 32, 34, 38].

2.2.1. Active Propellers

Kelsaidi et al. conducted a simulation study on an Underwater Swimming Manipulator (USM) intended for potential use in oil exploration. The USM design incorporated thrusters, in addition to actuated arm mechanisms, for maneuvering underwater. This robot merges a classic underwater autonomous vehicle together with an underwater snake robot, it is capable of serving as a manipulator while it is hovering underwater and so

performing eel-like motion when some application needs it, or in case of thruster's failure [59].

Gravdahl et al. worked on an articulated intervention autonomous underwater vehicle (AIAUV) called Eelume. This robot is equipped with thrusters that were strategically placed along the axis of the snake robot and in directions perpendicular to the axis, allowing the robot to generate thrust through serpentine movement when thrusters are not actuated, following a design strategy that is similar with respect to USM developed by Kelsaidi. This AIAUV was tested with different algorithms that try to properly integrate the thruster effect into trajectory control task, in order to address which strategy is optimal from an efficiency point of view [11].

Figure 2.3: Picture of the Eelume AIAUV [11]

2.2.2. Passive Propellers

Hirose et al. developed the ACM-R5, an amphibious snake robot designed to have a specific weight close to that of water. This snake could dive into and move on the surface of the water, equipped with universal joints for 3D movement and passive wheels for terrestrial locomotion. The solid extrusions supporting the wheels also acted as dorsal fins, contributing to its motion. The ACM-R5 achieved velocities of approximately 0.4 m/s both on land and in water [15, 26].

Mamba, initially conceived as a terrestrial snake robot, was later modified for amphibious movement. Modular in nature, experiments with varying numbers of links revealed that increasing the links from 10 to 20 resulted in a 20 % boost in forward velocity. The

snake, powered and controlled by a cable, demonstrated noteworthy performance. When equipped with a caudal fin as passive propellers, it achieved impressive speeds of 198.3 mm/s and 177.3 mm/s in lateral undulation and eel-like motion, respectively [34, 35, 38].

Figure 2.4: Underwater snake robot Mamba [34, 35, 38].

Ma et al. worked on an amphibious snake robot with 2-servo motors acting as universal joints, performing 3D movement. This robot, measuring 1.74 m in length with modules featuring four passive wheels, underwent experiments with various process parameters affecting its speed [70].

2.2.3. Propeller-less

Crespi et al. introduced AmphiBot 1, a modular amphibious snake robot capable of 2D motion without wheels, designed to be slightly buoyant. It achieved a maximum speed of around 35 mm/s on the water's surface [18]. AmphiBot 2, an improved version, allowed modules to connect without soldering, featured passive wheels, and included a water detector to address potential leaks. With more powerful motors, it achieved velocities of about 400 mm/s on land and 230 mm/s in water [17].

(a) AmphiBot 1. (b) AmphiBot 2.

Figure 2.5: Modular snake robots AmphiBot developed by Crespi et al. [17, 18]

Jasni et al. developed Snakey, an amphibious snake robot designed for surface swimming. It attained maximum velocities of approximately 1.2 mm/s on land and 3 mm/s in water.

However, due to its cable-based power and control system, its operational range was limited, and navigation was challenging due to the absence of sensors other than a camera [32].

Wright et al. conducted extensive research on snake robots, culminating in a waterproof model. This latest iteration featured eight modules powered by custom-designed servo motors. Communication between modules was facilitated using the RS485 protocol. To waterproof the robot, they applied various materials like nylon, polyester, and microfiber as a protective skin. However, this skin led to heat-related issues for the snake modules [66].

Additionally, Wright et al. developed another amphibious snake robot named Unified Snake, comprising 16 modules. Weighing approximately 3 kg, it measured 94 cm in length with a diameter of 5.1 cm. This snake robot was equipped with brakes that allowed it to maintain a position without consuming continuous energy. The brake only consumed energy during activation and deactivation, offering an energy-efficient holding mechanism. Despite being fully enclosed, humidity sensors were employed to detect potential water leaks. It is worth noting that the lack of airflow limited the effectiveness of these sensors. Communication with external computers was established using the RS-485 serial protocol. Although these robots were waterproof, there was no evidence indicating their use in swimming applications [67].

The most recent innovation in propeller-less robots was made by Robin Thandiackal et al. that have developed AgnathaX, a modular robot composed of 10 servomotors, batteries, a dedicated circuit, left and right antagonistic muscles as well as a pair of lateral force sensors for each module, all covered by a waterproof swim suit. The main feature of this work consists in the utilization of distributed force feedback loops for pattern generation [60].

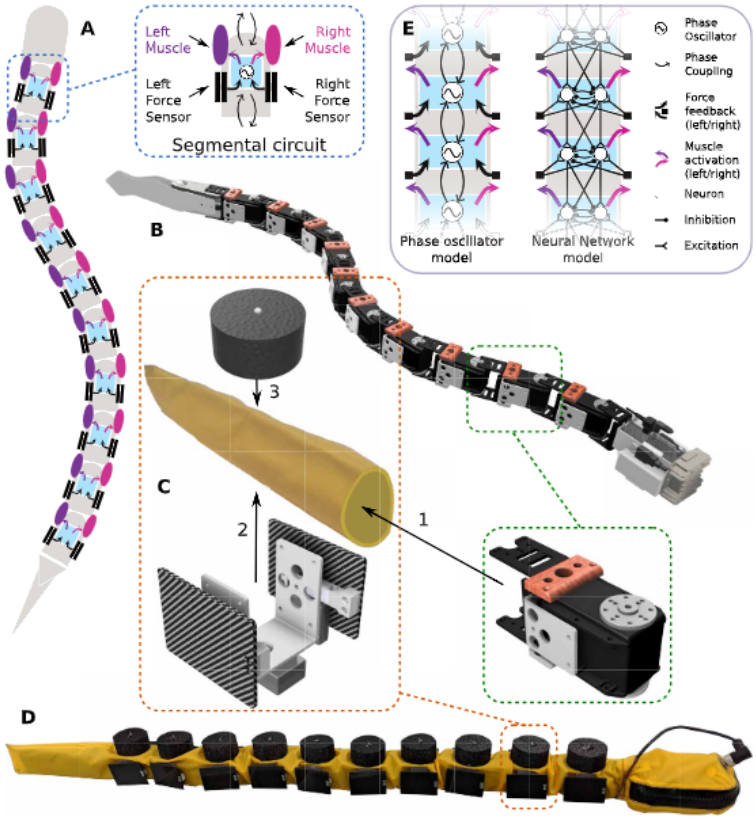


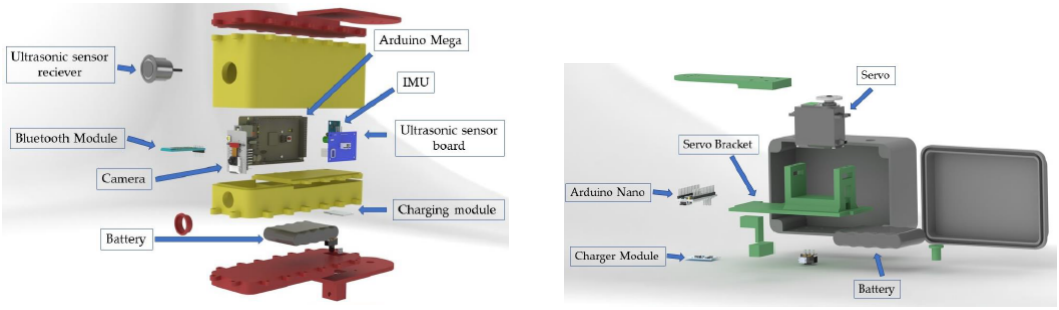
Figure 2.6: Modular snake robot AgnathaX [60].

3 | First Version of the Robot

The first thesis' objective was to optimize the performances of the snake robot previously built in PoliMi within the frame of a master's thesis [49]. Its structure is composed of a head, followed by eight modules, enclosed in a waterproof polyethylene cover.

The head is equipped with a switch, a 2000mAh/6V battery and its relative charger module, a Bluetooth module, to receive the user's instructions, an Arduino Mega board, to calculate and transmit the motion law to the single modules, an ultrasonic sensor, to spot the presence other objects and their distance, a camera, able to take pictures, and an IMU providing accelerations and angular velocities.

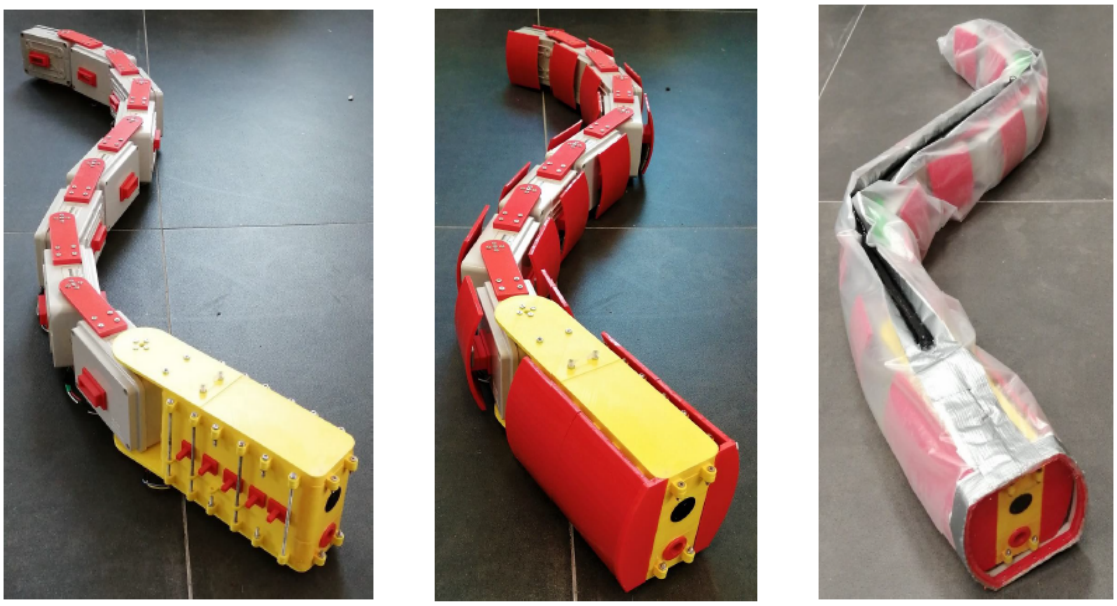
Each module is equipped with equal switch, battery and charger module. One Arduino Nano Every board is in charge to receive with I2C the instructions regarding the motion law and translate them to the servo motor, able to provide a maximum torque value of 1.5 Nm and a maximum speed of 6.54 rad/s.



(a) Head exploded view. (b) Module exploded view.

Figure 3.1: CAD exploded views of the head and module of the first version of the robot.

To modify the buoyancy level, the entire structure was ballasted. Finally, a polyethylene cover was introduced in order to make the structure waterproof.



(a) First robot assembled. (b) Added ballasts. (c) Final design.

Figure 3.2: First version of the robot.

3.1. Emerged Issues

From the tests performed in water on the first robot, many limitations emerged.

The Cover The first strong limitation consists of the thin polyethylene cover surrounding the robot. The water is able to pass through non perfect closures, causing catastrophic damages to the electrical parts and making the robot progressively sink. Moreover, the rigid nature of the cover prevents it from being adherent: as a result, a relative motion arises between the moving joints and the almost still cover, preventing an efficient motion transmission to the water outside, severely affecting the swimming performances.

Figure 3.3: First version of the robot tested on the field.

The Motion Transmission Each servomotor is housed in a plastic frame and screwed, transmitting the motion to the following module by using a plastic servo arm. After the first test, many frames were damaged, allowing a relative motion between the servomotor and the base. The servo arms were found deeply consumed after a few test, forcing the screw to carry the entire motion: with a slightly loose screw, the target is irretrievably lost.

The Weight Distribution Many ballasts are present and placed far from the center of mass of each module, to stabilize the entire structure and increasing its low sinking level caused by the air contained in the empty volume of the modules and inside the cover. This inefficient weight distribution results dramatically increases the inertia each servomotor has to carry on.

The Shape and the Robustness Each module is composed of a drilled electrical junction box, with two ABS 3D-printed lateral parabolic shapes glued to the surface and two thin arms screwed to the frame connecting to the following module. This solution forced two consecutive modules to be very distant, increasing the total length of the robot. Moreover, after the first test, the 3D-printed connection between the head and the first module was found delaminated.

The Switch-on of the Robot Each module has its own circuit, with its own switch. To turn on the robot, all the switches needed to be manually triggered. However, being

them placed inside the opaque cover, this operation is critical to be performed and quite time consuming.

The power provided by the batteries Due to the motors' needed update, it was evident that the previous batteries were not able to withstand a high electrical power consumption for extended periods: updating of the batteries became essential.

The head limitations Dealing with the head, many limitations emerged. Beyond sharing the same issues of the modules related to inertia properties, the closing mechanism consists of 28 stainless steel bolts, making any modification incredibly time consuming. Moreover, it was evident that the video-camera performances were inadequate: the output consists of some pictures taken at a very low frequency, impossible to be used to edit a video. The mounted ultrasonic sensor is not waterproof making it a high-risk component due to the water leakage problems encountered by the external cover. The data transmission mechanism is fully reliant on the Bluetooth module, causing unavoidable data losses each time the connection is temporarily lost.

3.2. The circuit

The old version's circuit is shown in Figure 3.4 and Figure 3.5, as one can notice the one of the module is very simple, while the one of the head is complex and it has components that do not justify the complexity of their programming, as for example the camera, which does not give a resolution capable to make the video useful for experiment. In the initial stages, our decision involved omitting those components and directing our efforts toward enhancing the architecture to prioritize more essential features.

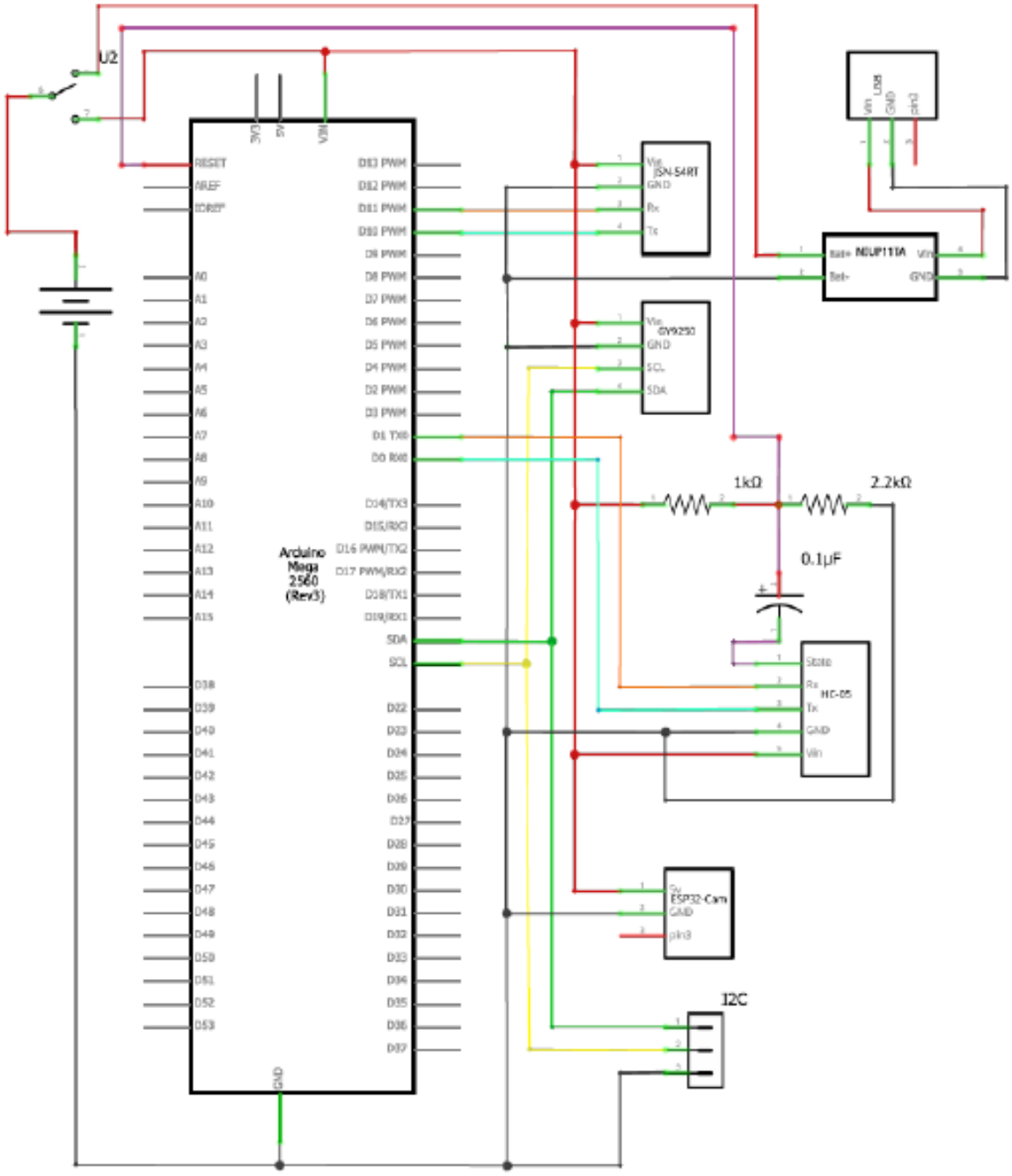


Figure 3.4: Previous circuit scheme of the head

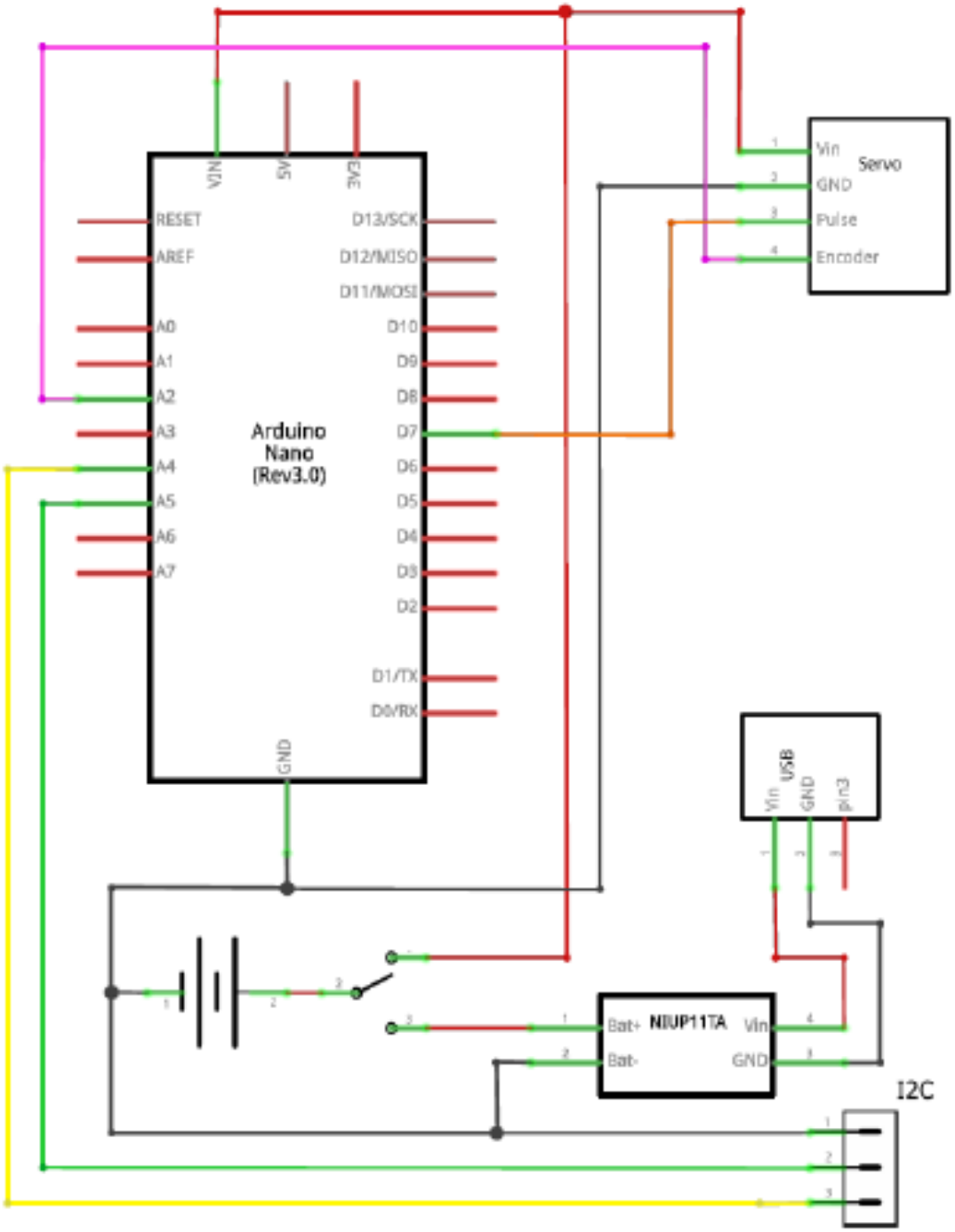


Figure 3.5: Previous circuit scheme for the modules

4 | Design of the New Version of the Robot

This chapter exposes the main solutions of the issues emerged with the first version of the robot, that led to the realization of the second, improved version.

4.1. Waterproofing

At first, a different cover mechanism was considered. After a meeting with the near company Dry Suit Experience [1], specialized in the realization of diving suits, it was evident that the need of frequent opening operations, dictated by the prototype nature of the project, were conflicting with the waterproofing and the strength level needed. For this reason, the cover component was abandoned and it was decided to make each module waterproof.

The cover removal made the drilled electrical boxes impossible to waterproof due to the different components junction. Instead, it has been decided to entirely 3D-print the whole robot's structure with the Zortrax M200 shown in Figure .

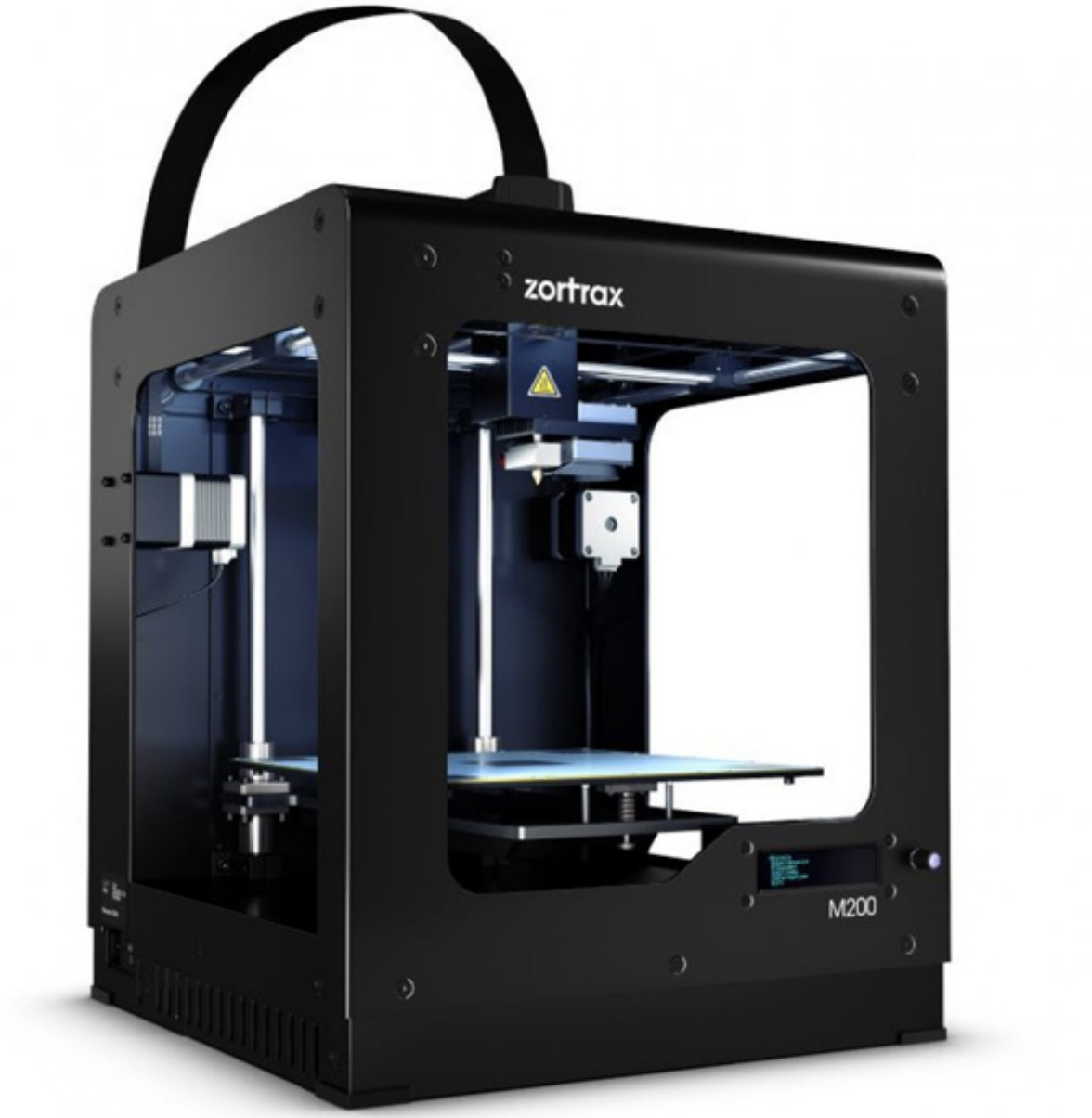


Figure 4.1: Zortrax M200 3D-printer

The 3D-printed ABS structure, due to the nature of the printing process, consisting of several filaments placed next to each other, lets water infiltrate. To solve this problem, it was decided to coat the external surface of the module with a chemical mixture of ABS and acetone, that melts the plastic filaments, creating a homogeneous waterproof surface cover.

Since each module is composed of multiple parts, each junction among them is designed watertight. The watertighteness is obtained through a dielectric gel, poured into a dedicated accommodation. The closure is performed while the gel is still liquid, to allow it to fill perfectly the joints interface. As a further precaution against water damages on the

circuit and to maximize the lateral surface interacting with water by making the robot sink more, the core of each module is filled with the dielectric gel. The cover dismission led to the integration of the IP68 connectors into the module, in substitution of the simple cables connecting the old version's modules. For further precaution against water's damages, each screw's head and accommodation was filled with dielectric gel or hot glue depending on the geometry.
The elimination of the polyethylene cover allows water to pass between the modules. For this reason, to maximize the force applied to the fluid, it is necessary to reduce the distance between two consecutive modules: the designed distance is set to 1 mm.

4.2. The Motion Transmission among Modules

The plastic servo arm was substituted by a more resistant aluminum one. Instead of just screwing it on the principal frame of the module, the motion transmission is achieved by a shape coupling, in addition to the M3 screw that is tightened to the servomotor on the following module.

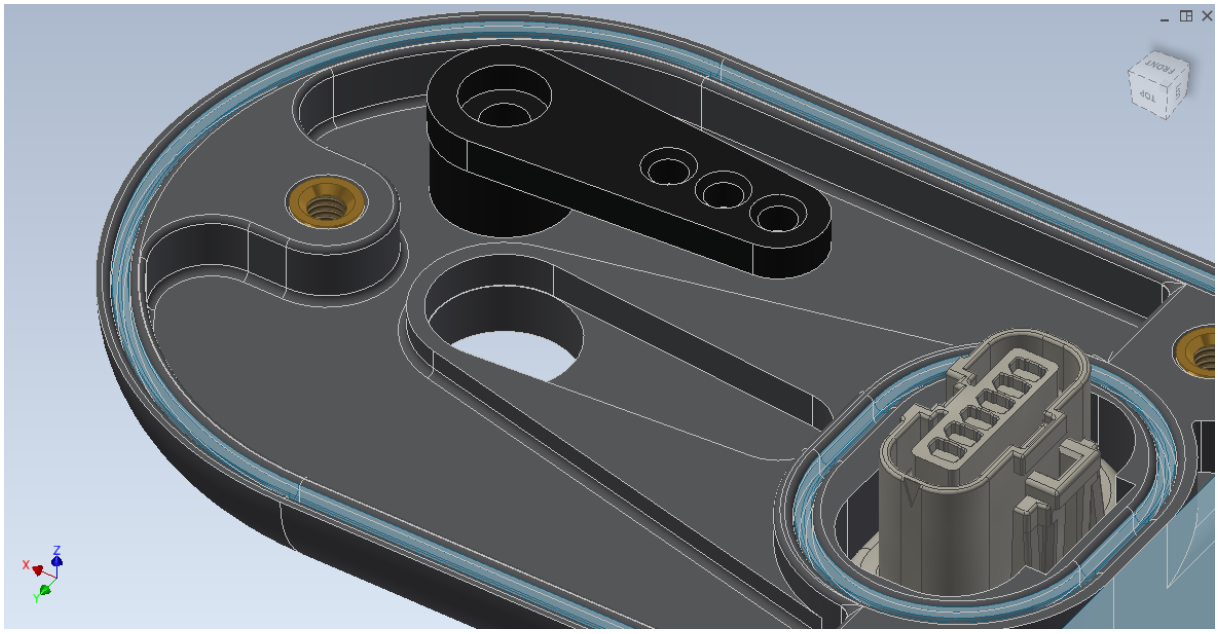


Figure 4.2: Shape coupling between the module's frame and the aluminum servo arm

The stator of the servomotor is kept fixed by a shape coupling between the module and the servomotor itself. Moreover, to improve locking, four flat-headed M3 screwed were used, in combination with brass threaded inserts for 3D-printed materials. To ensure that each module behaves as a rigid body, it was decided to design the principal frame as a

single component containing both the housing of the following aluminum servo arm and the current servomotor's stator.

4.3. Circuit

4.3.1. Circuit requirements

In the present version of the circuit, practical issues observed in the prior iteration have prompted modifications in the current iteration. Key aspects that warrant emphasis include: switches, charging procedures, incorporation of clear LED indicators, ultrasonic sensor reconsideration, and the integration of an SD module for data storage.

- Dealing with multiple switches could prove intricate when the robot experiences disconnections during practical tests. To address this, a configuration encompassing a single master switch for shutting down all modules and the head has been adopted, this makes the user capable of turning off the robot rapidly when an emergency occurs.
- The charging process was characterized by time inefficiencies and reliance on a charger with a complex form factor. The adoption of Li-Po batteries to enhance power and autonomy necessitated precise management of current and voltage during charging. Due to the sensitivity of Li-Po batteries to the charging process, components could not remain connected to the circuit during charging. This precaution was taken to prevent battery damage and extend its lifespan.
- Introduction of LED indicators is imperative to provide visual cues regarding the operational status of the hardware components.
- The replacement of the ultrasonic sensor arises from its inability to detect obstacles at close range. Although its theoretical minimum detection range stands at 20 cm, practical limitations, such as water interference, reduce its effective minimum range to approximately 1 m due to increased wave propagation speed in water.
- The inclusion of an SD module is pivotal for achieving enhanced connection stability. In the prior iteration, data transfer occurred via Bluetooth in real time, a practice that led to potential connection instability and unwarranted communication slowdowns. This was attributed to the sheer volume of data being transmitted, which did not significantly contribute to real-time user experience.

4.3.2. Electronic components

In this section, all electronic components that have been used are described.

- PowerHD 40 waterproof servomotors were selected both for their capability of providing higher torque and for the fact that they are IP68 certified. They are integrated with a microcontroller and they do not provide any feedback information. For this reason, we needed to integrate the system with external encoders.

	Value	Unit
Torque (7.4V)	$3,43$	Nm
Voltage	$6,0 - 8,4$	Hz
Mass	82	g
Velocity	60	rmp
Dimentions	$40,7$ x $20,5$ x $38,5$	mm

Table 4.1: Data of servomotors

- AS-5600 encoders were placed on the motor axis inside each module.

	Value	Unit
Type	Magnetic	/
Voltage	$3,3$	V
Resolution	$0,088(12bits)$	°
Signal	Analog	/

Table 4.2: Data of encoders

- HC-05 Bluetooth module, has the capability to work both as a master and a slave, the baudrate was set so to match the reprogramming specific of the Arduino board.

	Value	Unit
Voltage	5	V
Baudrate Used	115200	/
Data bit	8	/

Table 4.3: Data of HC-05 module

- SD-card reader for data saving.

	Value	Unit
Voltage	5	V
Maximum memory supported	16	GB
Communication protocol	SPI	/

Table 4.4: Data of SD-reader

- MPU-6050 IMU sensor is placed in the upper part of the head, it is connected to the Arduino Mega board through I^2C protocol, it has 3-axes accelerometer and a 3-axes gyroscope that we used to get information about the robot head's orientation.

	Value	Unit
Voltage	5	V
Accelerometer full scale	$\pm 78,5$	ms^{-2}
Accelerometer resolution	0,0024	ms^{-2}
Gyroscope full scale	$\pm 17,4$	$rads^{-1}$
Gyroscope resolution	$5x10^{-4}$	$rads^{-1}$
Communication protocol	I^2C	/
Mass	20	g

Table 4.5: Data of IMU

- OVONIC Lipo battery was chosen due to the capability of providing more current with respect to the previous one, as well as having a more compact structure with respect to NiMH batteries.

	Value	Unit
Type	LiPo	/
Voltage	7,4	V
Number of cells	2	/
Maximum current	110	A
Mass	100	g
Dimentions	18x33x87	mm

Table 4.6: Data of Batteries

- HC-SR04 Ultrasonic sensor is able to provide the required range of measure, making our robot capable of detecting obstacles from a shorter distance. The sensor is IP67-certified, but having it placed into the head we can ensure complete impermeability by the presence of gel used for isolating electronics.

	Value	Unit
Type	HC-SR04	/
Voltage	$3,3 - 5,0$	V
Range of measure	$3,0 - 450,0$	cm
Mass	44	g
Dimentions	64x30x19	mm

Table 4.7: Data of Ultrasonic sensor

4.3.3. Circuit design

All the issues illustrated in the previous circuit were addressed while formulating the design for the new version.

Starting with the switching-on challenge, our approach involved integrating a low-voltage relay that operates at 5V, functioning as a switch. This relay was connected to the Arduino Mega's 5V supply, and a relay was incorporated within each module. Through this setup, we achieved the capability to keep each circuit deactivated until the Arduino Mega in the head is turned on, by switching the physical switch on the robot's head.

The resolution to the charging issue entailed the introduction of a high-voltage relay, working at 8V, and responsive to the current flowing within the circuit during charging.

As the charger linked to the robot was engaged, the current split into two pathways. One pathway directed current towards the relay, while the other directed it directly to the battery for charging. The segment of current routed through the relay triggers circuit shutdown, ensuring that no other component remains connected to the battery during the charging phase. Additionally, a Schottky diode was introduced to avert reverse current flow. The selection of this diode was based on its low average voltage drop during operation, rendering the circuit more efficient.

For enhanced customization and better communication of the code's actions to the physical environment, four LEDs were integrated into the robot's head. Furthermore, one LED was linked to the built-in LED to monitor the operational status of the Arduino Mega.

Regarding the ultrasonic sensor, our preference was for the HC-SR04 model equipped with a waterproof ABS cover, which can detect obstacles with a minimum proximity of 2 cm. Accounting for the propagation speed of waves in water, this configuration effectively results in real-time detection at a distance of 10 cm, satisfying our requirements.

To cater to data retention for analysis purposes and to prevent data loss during experimental tests, an SD module was also implemented within the robot's head.

All the aspects under discussion are visually represented within Figure 4.3, which pertains to the head circuit, and Figure 4.4, which focuses on the modules.

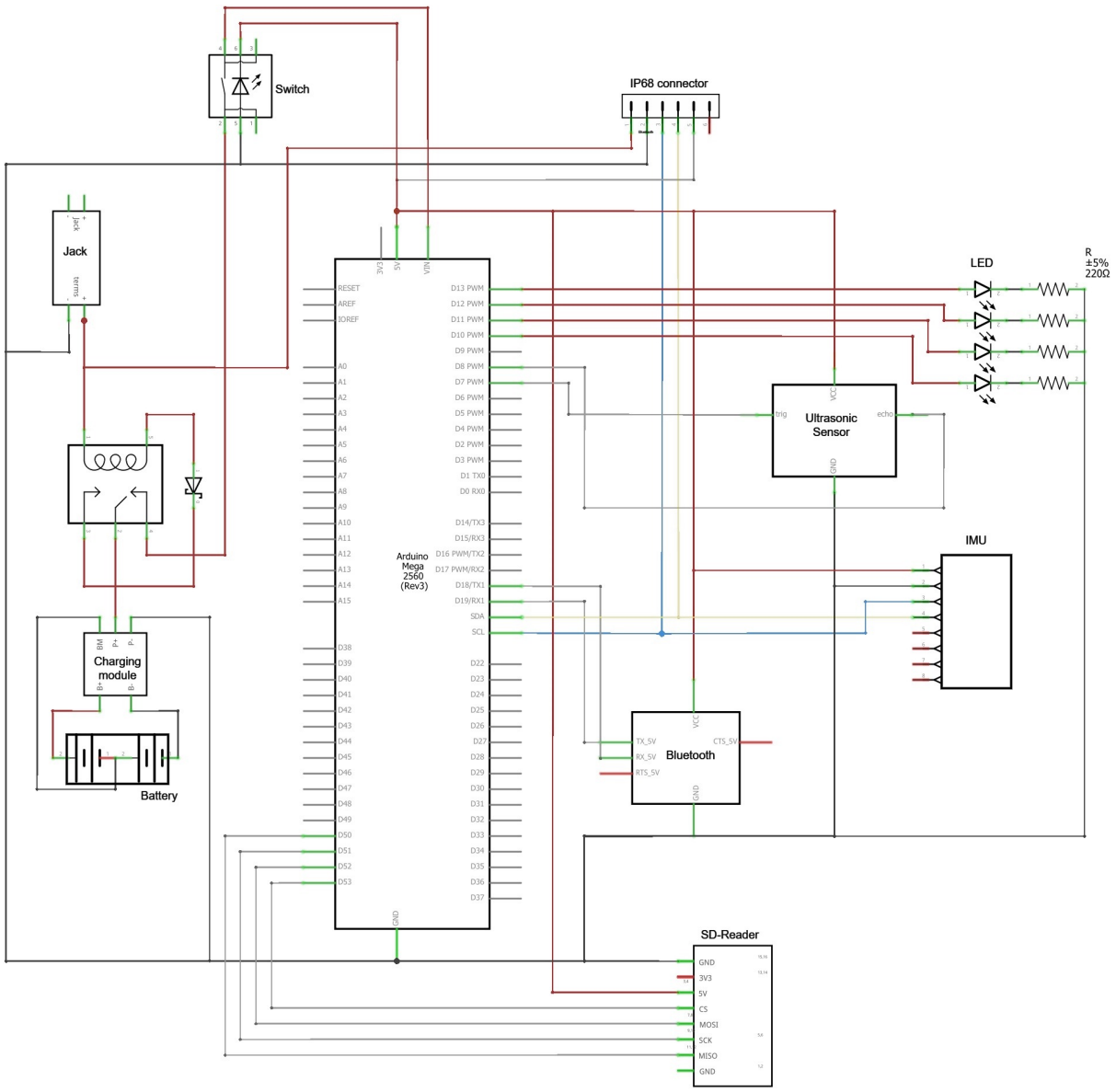


Figure 4.3: New circuit scheme of the head

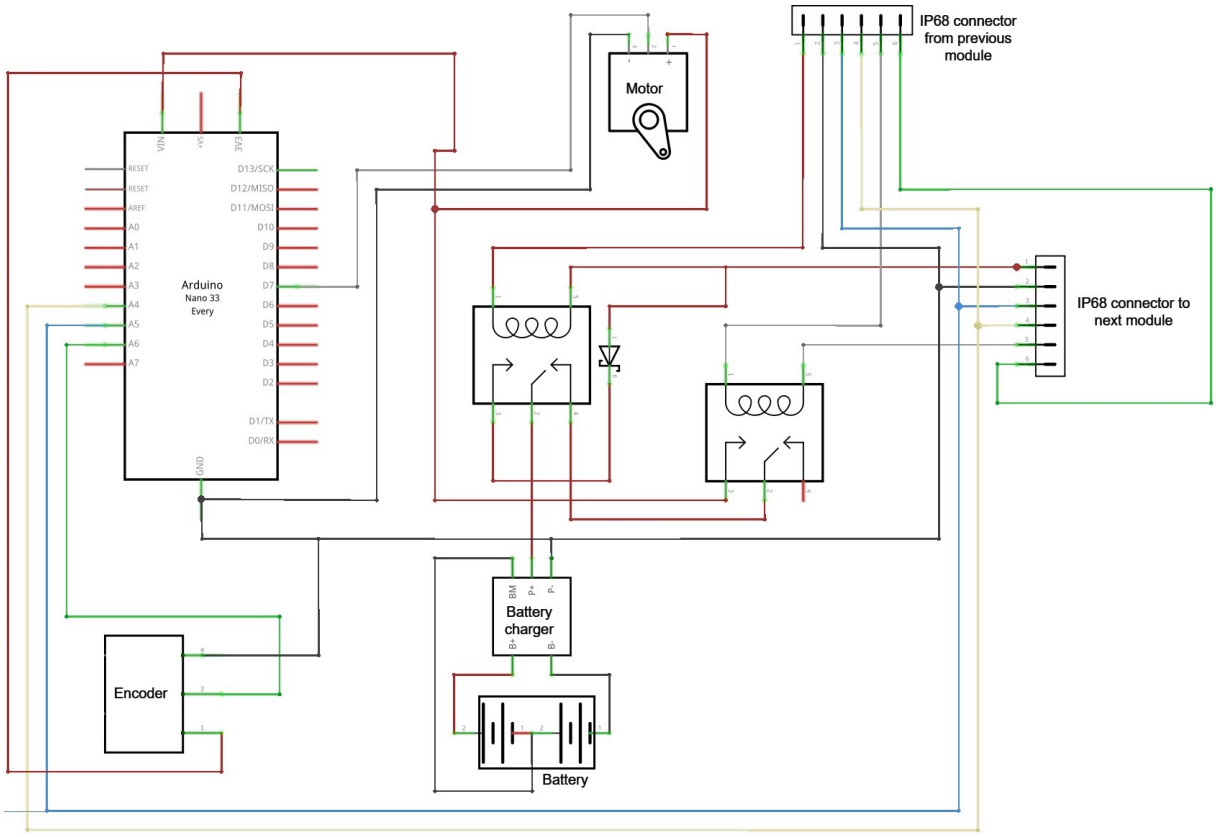


Figure 4.4: New circuit scheme for the modules

4.4. The Modules

The cover removal forced to 3D-print the entire structure of the robot, allowing a complete re-design of every single component, performed using Autodesk Inventor. As stated in Section 4.2, to enforce the strength of the motion transmission, it was decided to keep the housing of the following aluminum servo arm and the current servomotor's stator in the same structure.

The distance between this two axes defines the length of the entire robot. The complex external shape is dictated by the need of minimizing the amount of water able to pass between two consecutive modules while ensuring a robust movement. The prismatic volume was abandoned in favour of this 3D buttonhole shape. The two semi-circumferences allow to minimize the distance between two consecutive modules, maximizing the thrust surface associated to a given total length. The first semi-circumference is centered on the servomotor's axis.

Knowing the size of the servomotor, the battery, the connector and the electronics, the distance with the second semi-circumference is, thus, obtained. To minimize the robot's

length, the battery is positioned vertically, adjacent to the servomotor: this choice dictates the vertical dimension of each module. An extrusion containing the housing of the servo arm was created, in order to obtain a tolerance of 1 mm between two consecutive module's external surfaces both in horizontal and vertical directions. To avoid out of plane motion, this extrusion was mirrored in the vertical direction, with the addition of a plastic pin; the consequent hole is then created on the servomotor's axis creating a cylindrical joint.

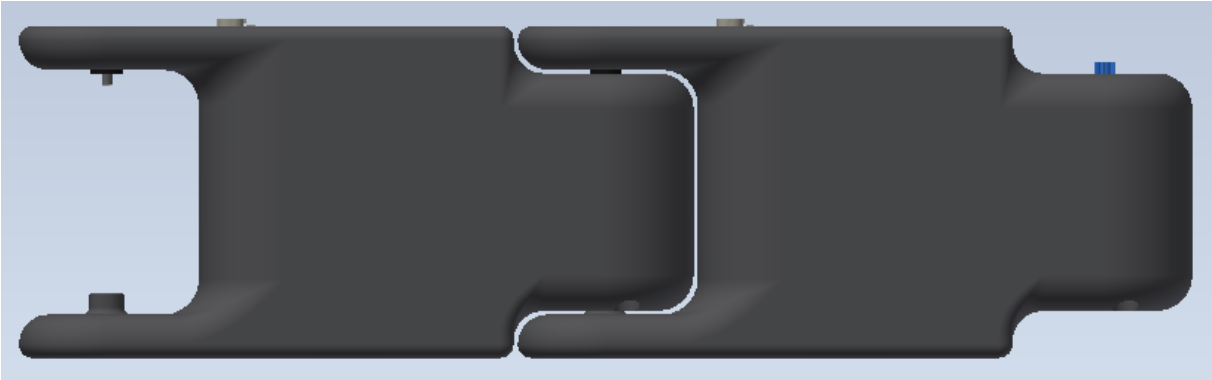


Figure 4.5: Cylindrical joint between modules

To improve the robot's control, an external magnetic encoder is added to each module. A magnetic cylinder is housed externally in the plastic pin, while the read-head is positioned inside the module, below the motor's frame. To allow the data reading, considering this magnet model, the vertical maximum distance between these two components is about 1.5 mm: for this reason, strict vertical tolerances are applied in order to guarantee a stable feedback signal.

4.5. Buoyancy

Once it was decided to 3D print the whole shape, an important decision regarding printing parameters had to be made. In order to make the overall structure float we evaluated the hydrostatic force according to Archimedes' principle

$$F_p = F_a \tag{4.1}$$

$$m_m = V_w \rho_w \tag{4.2}$$

The module's mass m_m can be evaluated as the sum of different contributions

$$m_m = V_{abs}\rho_{abs}\bar{\rho} + m_s + m_{arm} + m_b + m_{el} + m_{ballasts} \tag{4.3}$$

that are shown in Table 4.8, where $\bar{\rho}$ is a coefficient representing the print density, while V_w can be simply evaluated from the CAD as the total volume occupied by the module.

	Description	Value	Unit
V_{abs}	Volume of the ABS structure	$3,414 \cdot 10^{-4}$	m^3
ρ_{abs}	Density of ABS	$1,040 \cdot 10^{-3}$	$\frac{kg}{m^3}$
m_s	Servomotor's mass	$8,730 \cdot 10^{-2}$	kg
m_{arm}	Aluminum arm's mass	$1,401 \cdot 10^{-3}$	kg
m_b	Battery's mass	$1,190 \cdot 10^{-1}$	kg
m_{el}	Electronic components + gel mass	$1,170 \cdot 10^{-1}$	kg
m_b	Ballasts' mass	$7,000 \cdot 10^{-2}$	kg

Table 4.8: List of parameters used for Buoyant force evaluation.

Knowing that having perfect balance can be risky, we adjusted the value of V_w considering that just 95% of the height of the module is immersed in the fluid, in order to prevent the overall robot from sinking. At this point, it is possible to re-arrange Equation 4.3 to calculate the right value for print density in order to obtain the desired level of buoyancy

$$\bar{\rho} = \frac{V_w\rho_w - (m_s + m_{arm} + m_b + m_{el} + m_{ballasts})}{V_{abs}\rho_{abs}} = 0,20 \tag{4.4}$$

4.6. The Last Module Differences

With respect to the previous modules, the last one present substantial differences. Being the last in the chain, the arms containing the aluminum servo arm and the magnetic cylinder have no purposes, as well as the IP68 connector. These components were removed and the module was modified to include the attachment of the silicone tail, that will be introduced in future work to increase the thrust of the robot.

4.7. Modules Realization and Assembly

The modules' shape is split into 4 parts named A,B,C and D. In Figure 4.7, the central part of the module (module A) contains the housing of the servomotor, the battery and circuit, the IP68 connector and the aluminum servo arm. To minimize any possible damage due to water leaking, the central module is divided into two chambers: the first one contains the servomotor while the second contains the battery, all the electrical parts and the connector. To face any specific issue to a component, it is possible to access separately one of the two chambers. The second part (module B) closes the servomotor's chamber and contains the housing for the magnetic encoder reader.
The third part (module C) consists of the upper cover of the whole module: by removing this part it is possible to access the chamber containing the battery, the electronic board and all the internal circuit.
The last part (module D) consists of the lower arm of the module, containing the housing of the magnetic cylinder and the ballasts. Each part is connected to the central part A with the usage of stainless steel countersunk M3 screws in combination with brass threaded inserts. All the inserts are positioned in the part A by the usage of the ABS-acetone mixture, increasing the mechanical resistance with respect to the simple interference coupling and guaranteeing the waterproofing of the component. The dimensions of the housings for all the components were singularly tested to ensure the desired behaviour, such as an effective interference coupling for the IP68 connector or a slightly clearance coupling for the battery and the servomotor. Regarding the dimensions of the holes for the brass threaded inserts, a single component with different diameters was printed, the inserts were positioned using the ABS-acetone mixture and then tested.

(a) 3D-printed specimen. (b) brass inserts specimen.

Figure 4.6: 3D-printed specimens.

The parts orientation on the printing bed were chosen to optimize the support removal and to guarantee the best possibly printing quality for high interest components, such as the housing of the encoder reader and the magnetic cylinder. Once the orientation were

chosen, the watertight extrusion were generated, to avoid any unwanted support generation. Regarding the printing of the sub-module A, a severe warping problem emerged. To solve this issue, the printing bed was covered with the ABS-acetone mixture, that effectively counteracted the thermal withdrawal.

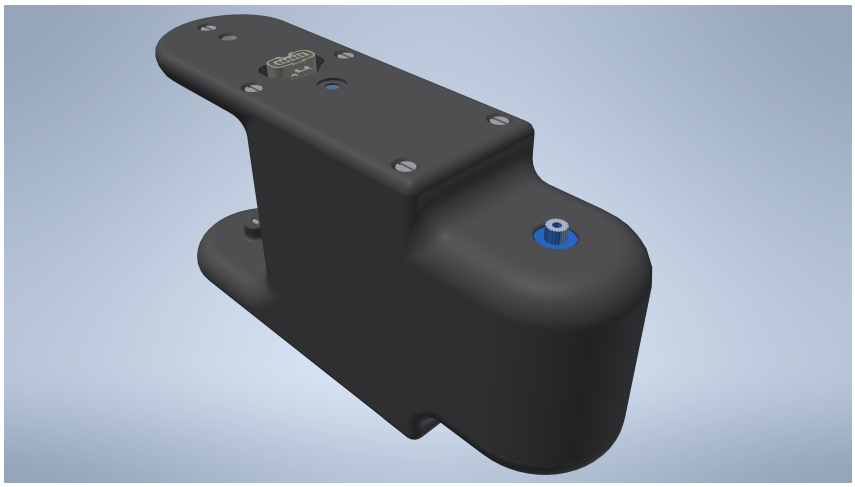


Figure 4.7: Module assembled

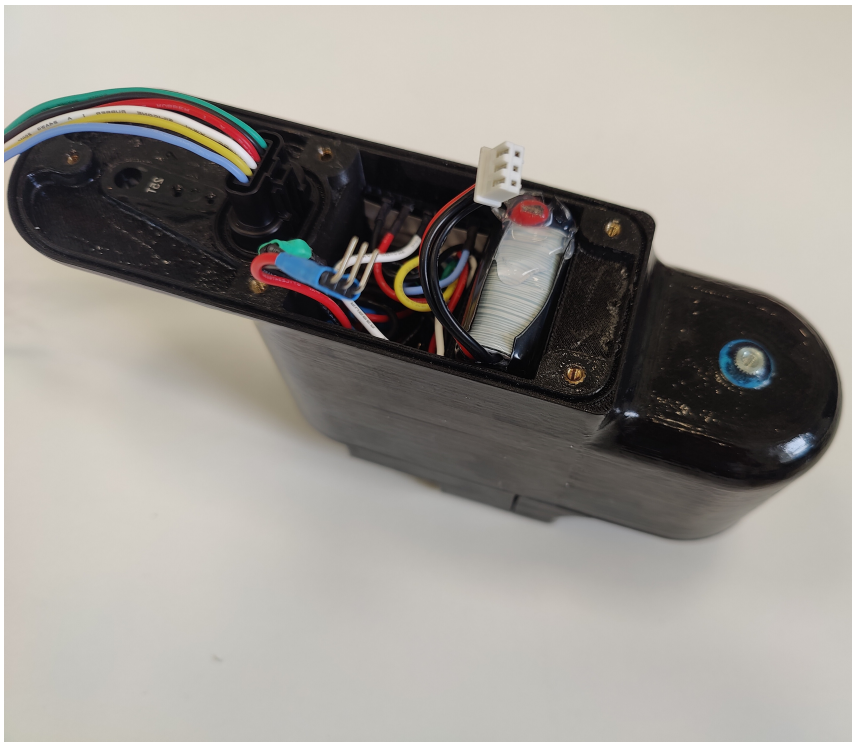


Figure 4.8: Module printed and assembled

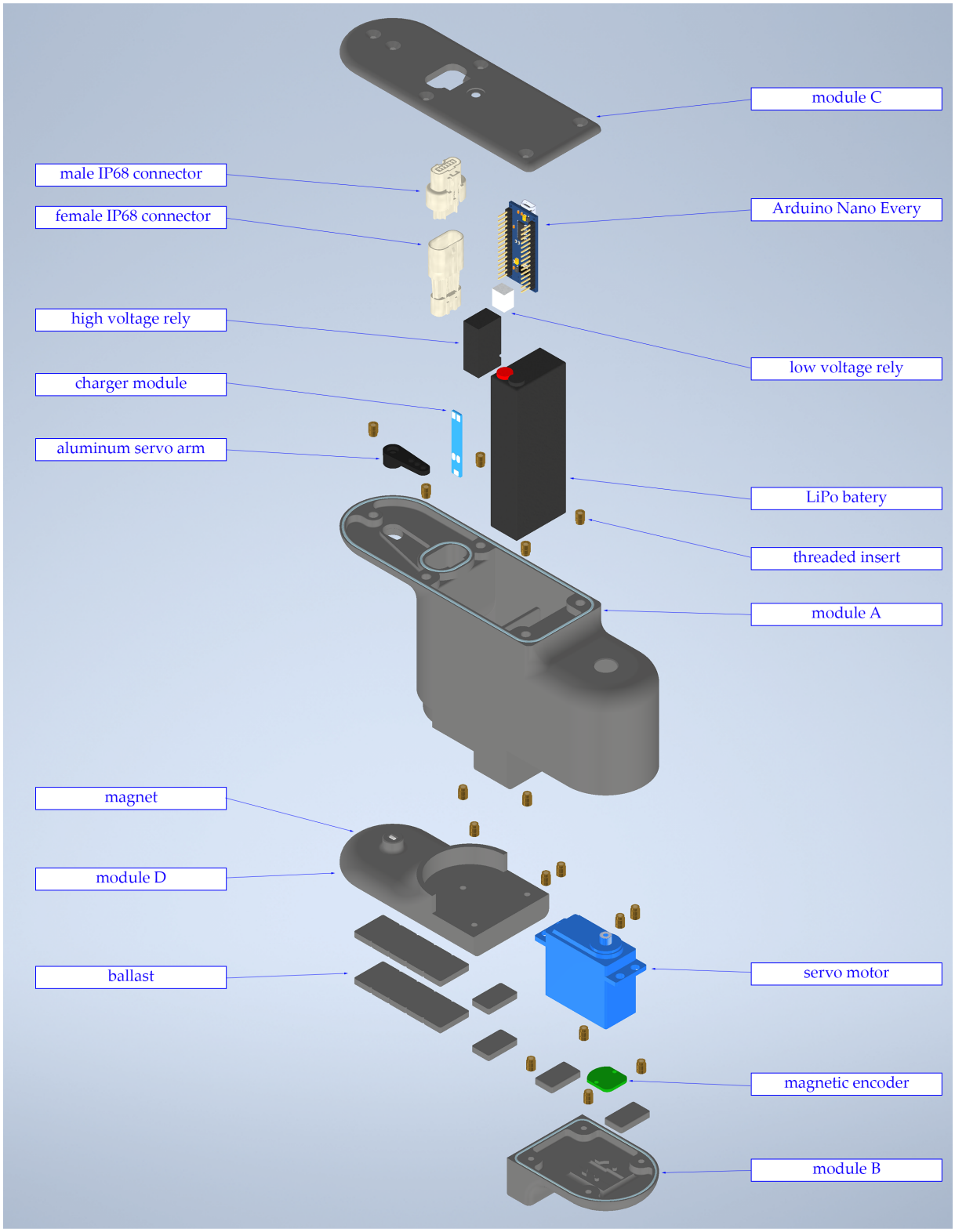


Figure 4.9: Exploded view of the module

4.8. The Head

In re-designing the head, many decisions regarding the components had to be made. The video camera was removed due to its inadequate output. Regarding the ultrasonic sensor, a different, waterproof model was chosen and hand-cut to reduce its dimensions and allow to minimize the head volume.
A micro-SD reader component is added to minimize the risk of data losses. One single, global, power button was introduced in the head, together with a single connector for battery charging. The shape was created by using the Autodesk Inventor FreeForm feature, that allows to directly 3D-sculpt the final external shape, that will then be CAD-edited to insert all the components housing. The vertical dimension as well as the attachment to the first module must be the same as those of the other modules: the propeller housing dimensions are already defined, as well as the IP68 interference coupling. To keep a similar side encubrance, the Arduino Mega board is positioned vertically, on the left side of the head, with the battery lying on the bottom of the shape. The micro SD reader is vertically positioned leaning against the IP68 wall, to guarantee a handy SD card removal each time is needed. In order to work properly, the ultrasonic sensor is positioned frontally, within its designed housing. The accelerometer is positioned on top of the upper part of the head, between the power button and the charging attachment with a strict interference coupling that prevents any internal movement that would interfere with the obtainable data. All the other components have no dedicated housing inside the head structure.

4.9. Head Realization and Assembly

As for the printing of the modules, the head is split in 3 parts: the lower head part (head A), the upper head part (head B) and the closing of the aluminum propeller (head C), as shown in Figure 4.12. Being the shape not constant in height, the splitting plane between the lower part (head A) and the upper part (head B) is moved down with respect to the other modules. To minimize the supports needed, the lower part of the head is printed upwards, with the usage of the ABS-acetone mixture to avoid warping defects: for this reason, the negative shape of the watertight extrusion is positioned in this part. Vice versa, to guarantee a high printing quality needed in the accelerometer's housing, the top part of the head is printed upside down. All the parts are connected together with the usage of six stainless steel countersunk M3 screws, together with the brass threaded inserts.

Figure 4.10: Head assembled

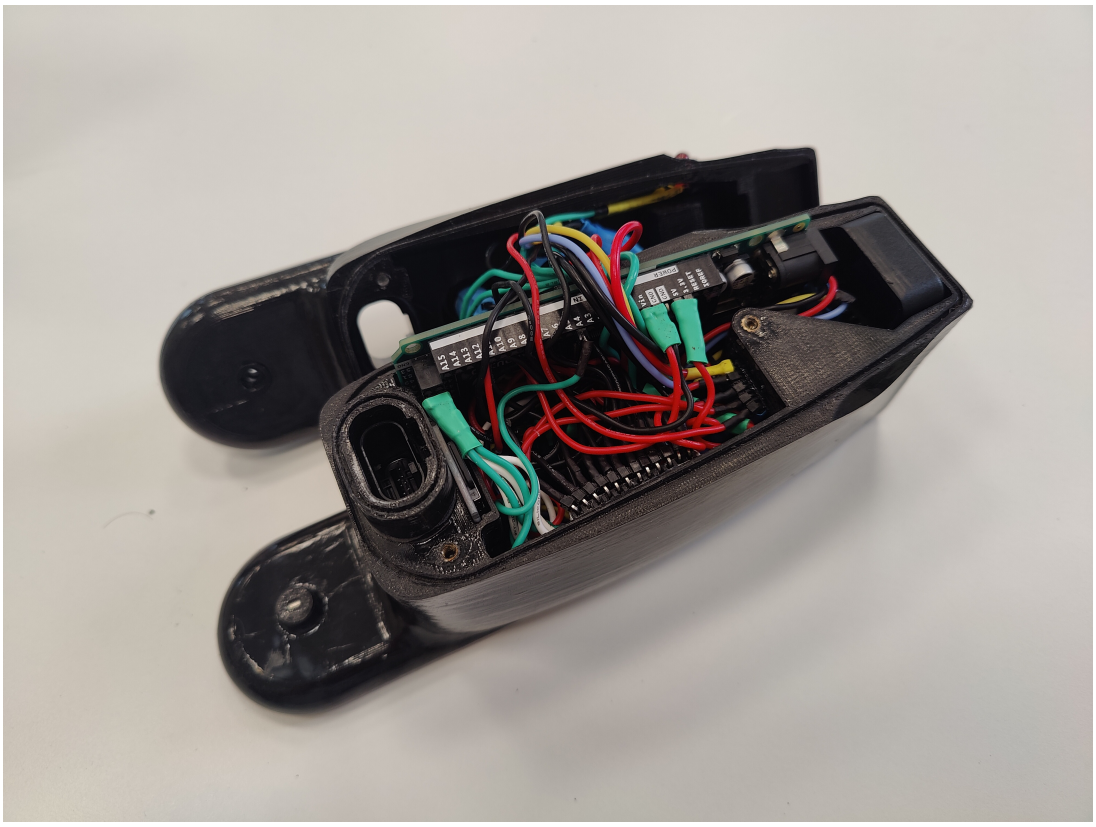


Figure 4.11: Head printed and assembled

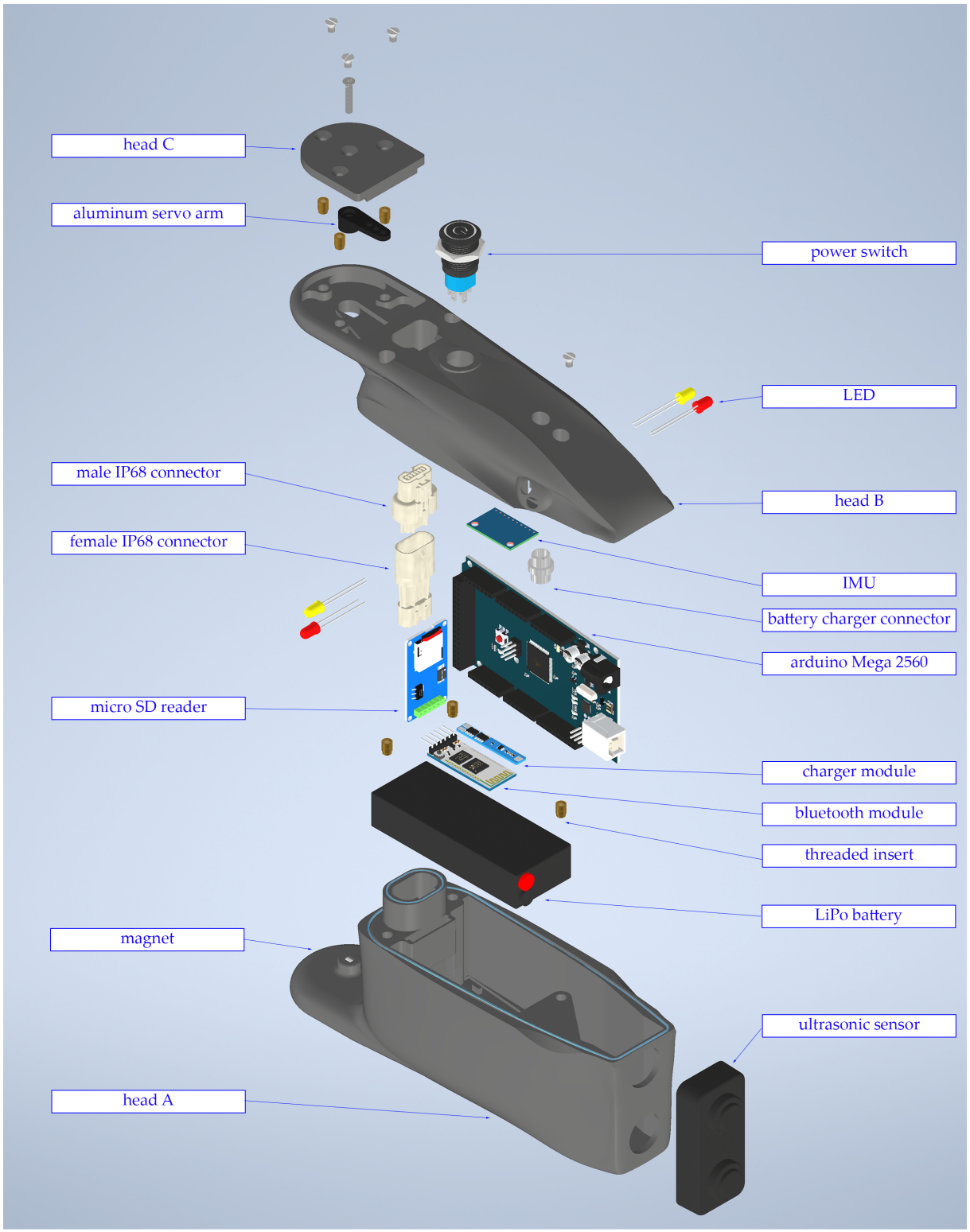


Figure 4.12: Exploded view of the head

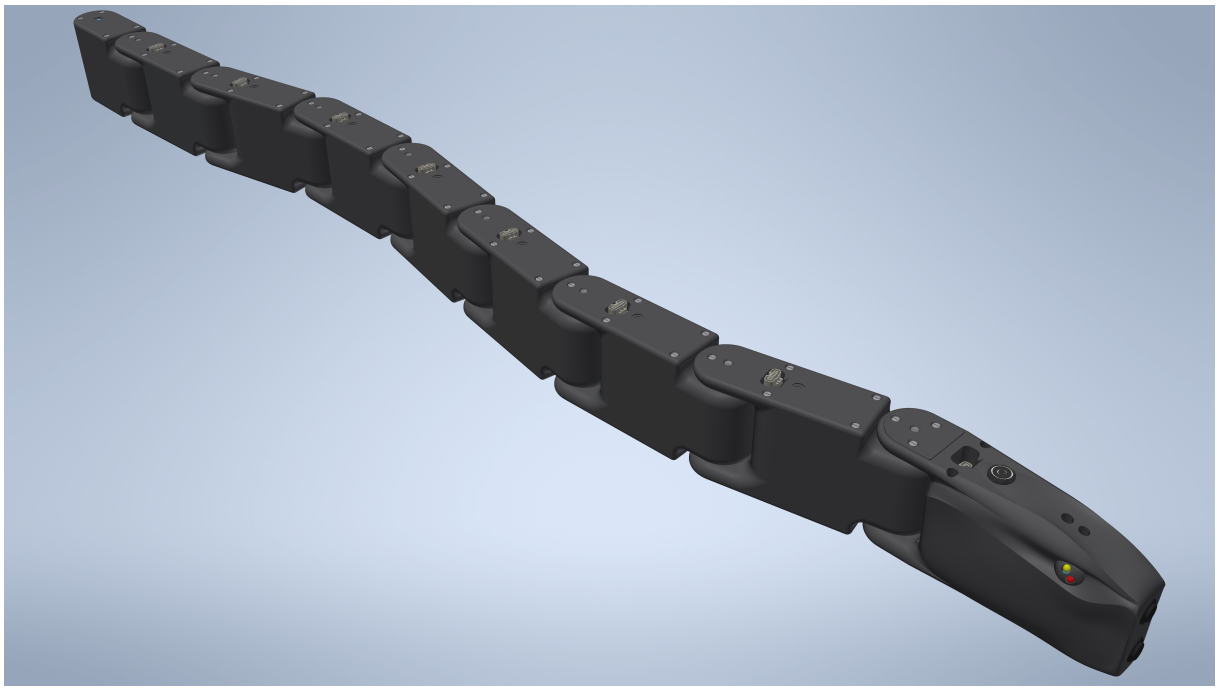


Figure 4.13: Final CAD assembly

In Figure 4.13, the final assembly of the robot is shown. The head weights 0.65 kg and is 204 mm long, while for the seven following modules these properties are 0.60 kg and 191 mm. The last module is slightly different, weighting 0.50 kg and being 140 mm long, for a total weight of the entire robot of 4.98 kg, an overall length, width and height of 1281 mm, 60 mm and 94 mm, respectively.

Figure 4.14: Assembled robot

5 | Simulation

The first objective of this thesis was to optimize the first version of the robot, understanding its limitations and highlighting which parameters would maximize its swimming performances. Those needs led to the introduction of a simulation able to reproduce the structure characteristics and inertial properties of the robot as well as rapidly testing the huge amount of motion law's parameters combination. Moreover, any modification of the robot design can be easily introduced, making it suitable to compare similar robot models or different design development solutions.

The simulation is created using Unity, a cross-platform game engine that allows to create interactive, real-time simulations through the usage of the *C#* programming language. The structure of the snake robot is created with the use of the Articulation Body class, able to build physics articulations such as kinematic chains with Game Objects that are hierarchically organized [5]. Among the numerous advantages with respect to the Rigid Body class [5], starting from the 2022.2.0f1 Unity version, it is possible to apply the inverse dynamics to the Articulation Body class. Within Unity it is possible to include code created outside of this ambient in the form of a plug-in [4]. The Moving Least Squared-Material Point Method algorithm used to simulate the water behaviour is introduced inside the simulation with the usage of the Zibra Liquids plug-in [6]. The enire simulation is executed with the usage of Unity's built-in 3D physics engine, which is an integration of Nvidia PhysX engine [3]. In order to correct simulate physics, it is important to call the function `MonoBehaviour.FixedUpdate()`, which is executed at the same frequency as the physics system, rather than `MonoBehaviour.Update()`, which is executed every frame rate and can be out of synchronization depending on the graphics load involved [2].
It must be stressed out that the choice of this algorithm is forced by the strict time limitations: test a sample of parameters combination with a more precise CFD approach would be prohibitively time consuming. The less accurate output of this approach is compensated by the possibility of simulating a great number of different motion laws to visual inspect the robot behaviour, evaluate a comprehensive global speed trend among the different combinations and the order of magnitude of the torques involved.

5.1. MPM-MLS algorithm

5.1.1. Introduction

The presented algorithm, consists of an hybrid approach, utilizing both meshless Lagrangian particles and a background Eulerian grid.

5.1.2. Kinematics Theory

As stated by [33], adopting the continuum assumption, each MPM particle is subset of the entire simulated material domain.

Continuum Motion

The motion of material is determined by a deformation map

$\phi(\cdot, t) : \Omega^0 \rightarrow \Omega^t \; for \; \Omega^0, \Omega^t \subset \mathbb{R}^d$

where d = 2 or 3 is the domain dimension. ϕ describes the motion of each point $\boldsymbol{X} \in \Omega^0$ over time. $\boldsymbol{X}$, represents points in the set Ω^0, also called material points. $\boldsymbol{x} \in \Omega^t$ express the location of material points at time t

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{X}, t) = \boldsymbol{\phi}(\boldsymbol{X}, t). \tag{5.1}$$

Velocity

$\boldsymbol{V}(\cdot, t) : \Omega^0 \rightarrow \mathbb{R}^d$

$$\boldsymbol{V}(\boldsymbol{X}, t) = \frac{\partial \phi}{\partial t}(\boldsymbol{X}, t) \tag{5.2}$$

and acceleration

$\boldsymbol{A}(\cdot, t) : \Omega^0 \rightarrow \mathbb{R}^d$

$$\boldsymbol{A}(\boldsymbol{X}, t) = \frac{\partial^2 \phi}{\partial t^2}(\boldsymbol{X}, t) = \frac{\partial \boldsymbol{V}}{\partial t}(\boldsymbol{X}, t) \tag{5.3}$$

can be quantified based on the Lagrangian view.

Deformation

Any point $\boldsymbol{X}$ in Ω^0 is mapped to Ω^t for a given time t via $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{X}, t)$.

The Jacobian of the deformation map ϕ, or deformation gradient, is a function $\boldsymbol{F}(\cdot, t) : \Omega^0 \to \mathbb{R}^{dxd}$

defined as

$$\boldsymbol{F}(\boldsymbol{X}, t) = \frac{\partial \phi}{\partial \boldsymbol{X}}(\boldsymbol{X}, t) = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}(\boldsymbol{X}, t) \tag{5.4}$$

The index notation can be used:

$$F_{ij} = \frac{\partial \phi_i}{\partial X_j} = \frac{\partial x_i}{\partial X_j}, \qquad i, j = 1, ..., d. \tag{5.5}$$

It expresses the material local deformation level. Its determinant represents infinitesimal volume change

$J = det(\boldsymbol{F})$.

Push Forward and Pull Back

As for [33], it is assumed that no two different particles of material ever occupy the same space at the same time. $\forall \boldsymbol{x} \in \Omega^t, \exists! \boldsymbol{X} \in \Omega^0$ such that $\phi(\boldsymbol{X}, t) = \boldsymbol{x}$. For this reason, there exist an inverse mapping $\phi^{-1}(\cdot, t) : \Omega^t \to \Omega^0$. Taking a function defined over Ω^0 and defining a counterpart over Ω^t is called Eulerian/push forward. On the contrary, taking a function defined over Ω^t and defining its correspondent over Ω^0 is referred as Lagrangian/pull back. The equations 5.2 and 5.3 define the velocity and acceleration as Lagrangian. They Eulerian counterparts are,

$$\boldsymbol{v}(\boldsymbol{x}, t) = \boldsymbol{V}(\phi^{-1}(\boldsymbol{x}, t), t) \tag{5.6}$$

$$\boldsymbol{a}(\boldsymbol{x}, t) = \boldsymbol{A}(\phi^{-1}(\boldsymbol{x}, t), t). \tag{5.7}$$

With the chain rule,

$$\boldsymbol{A}(\boldsymbol{X}, t) = \frac{\partial \boldsymbol{V}}{\partial t}(\boldsymbol{X}, t) = \frac{\partial \boldsymbol{v}}{\partial t}(\boldsymbol{\phi}(\boldsymbol{X}, t), t) + \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}(\phi(\boldsymbol{X}, t), t)\frac{\partial \phi}{\partial t}(\boldsymbol{X}, t). \tag{5.8}$$

It is possible to rewrite it adopting index notation

$$A_i(\boldsymbol{X},t) = \frac{\partial V_i}{\partial t}(\boldsymbol{X},t) = \frac{\partial v_i}{\partial t}(\boldsymbol{\phi}(\boldsymbol{X},t),t) + \frac{\partial v_i}{\partial x_j}(\phi(\boldsymbol{X},t),t)\frac{\partial \phi_j}{\partial t}(\boldsymbol{X},t). \tag{5.9}$$

where j implies summation.

Combining Equation 5.2, 5.6, 5.3 and 5.9

$$a_i(\boldsymbol{x},t) = A_i(\phi^{-1}(\boldsymbol{x},t),t) = \frac{\partial v_i}{\partial t}(\boldsymbol{x},t) + \frac{\partial v_i}{\partial x_j}(\boldsymbol{x},t)v_j(\boldsymbol{x},t). \tag{5.10}$$

thus,

$$a_i(\boldsymbol{x},t) \neq \frac{\partial v_i}{\partial t}(\boldsymbol{x},t). \tag{5.11}$$

Material Derivative

The notation

$$\frac{D}{Dt}v_j(\boldsymbol{x},t) = \frac{\partial v_i}{\partial t}(\boldsymbol{x},t) + \frac{\partial v_i}{\partial x_j}(\boldsymbol{x},t)v_j(\boldsymbol{x},t) \tag{5.12}$$

is introduced to obtain

$$\boldsymbol{a} = \frac{D}{Dt}\boldsymbol{v}. \tag{5.13}$$

For a generic Eulerian function $f(\cdot,t) : \Omega^t \to \mathbb{R}$ the same notation is used to mean

$$\frac{D}{Dt}f(\boldsymbol{x},t) = \frac{\partial f}{\partial t}(\boldsymbol{x},t) + \frac{\partial f}{\partial x_j}(\boldsymbol{x},t)v_j(\boldsymbol{x},t). \tag{5.14}$$

Being $\frac{D}{Dt}f(\boldsymbol{x},t)$ the push forward of $\frac{\partial}{\partial t}F(\boldsymbol{X},t)$ with $F(\cdot,t) : \Omega^0 \to \mathbb{R}$ being a Lagrangian function.
The same principle can be used for the deformation gradient, commonly defined as Lagrangian: $\boldsymbol{F}(\cdot,t) : \Omega^0 \to \mathbb{R}^{dxd}$.

$$\frac{\partial}{\partial t}F_{ij}(\boldsymbol{X},t) = \frac{\partial}{\partial t}\frac{\partial \phi_i}{\partial X_j}(\boldsymbol{X},t) = \frac{\partial V_i}{\partial X_j}(\boldsymbol{X},t) = \frac{\partial v_i}{\partial x_k}(\phi(\boldsymbol{X},t),t)\frac{\partial \phi_k}{\partial X_j}(\boldsymbol{X},t). \tag{5.15}$$

Consider $\boldsymbol{f}(\cdot, t) : \Omega^t \to \mathbb{R}^{dxd}$ as the push forward of $\boldsymbol{F}$, then

$$\frac{D}{Dt}\boldsymbol{f} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}\boldsymbol{f} \qquad or \qquad \frac{D}{Dt}f_{ij} = \frac{\partial v_i}{\partial x_k}f_k j \tag{5.16}$$

Volume and Area Change

Consider an infinitesimal volume $dV = dL_1\boldsymbol{e}_1 \cdot (dL_2\boldsymbol{e}_2 \times dL_3\boldsymbol{e}_3)$, with, $d\boldsymbol{L}_i = dL_i\boldsymbol{e}_i$.

In world space, it is possible to define

$$d\boldsymbol{l}_i = \boldsymbol{F}d\boldsymbol{L}_i. \tag{5.17}$$

Considering $dl_1dl_2dl_3 = JdL_1dL_2dL_3$,

for any function $G(\boldsymbol{X})$ or $g(\boldsymbol{x}, t)$, Eulerian/Lagrangian can be used to change variables for volume integrals defined for Ω^0 or Ω^t, as well as on any of their subsets:

$$\int_{B^t} g(\boldsymbol{x})dx = \int_{B^0} G(\boldsymbol{X})J(\boldsymbol{X}, t)d\boldsymbol{X}, \tag{5.18}$$

with B^t arbitrary subset of Ω^t.
Similarly, considering an arbitrary infinitesimal area dS in Ω^0 with its normal $\boldsymbol{N}$, their Eulerian counterparts are defined as ds and n.

$$d\boldsymbol{S} = (dS)\boldsymbol{N}, \tag{5.19}$$

$$d\boldsymbol{s} = (ds)\boldsymbol{n}. \tag{5.20}$$

Considering an infinitesimal vector $d\boldsymbol{L}$ and its deformed version $d\boldsymbol{l}$,

$$dV = d\boldsymbol{S} \cdot d\boldsymbol{L}, \tag{5.21}$$

$$dv = d\boldsymbol{s} \cdot d\boldsymbol{l}. \tag{5.22}$$

Combining the Equations 5.21, 5.22 and the relation $dv = JdV$:

$$Jd\boldsymbol{S} \cdot d\boldsymbol{L} = d\boldsymbol{s} \cdot (\boldsymbol{F}d\boldsymbol{L}), \qquad with \qquad d\boldsymbol{l} = \boldsymbol{F}d\boldsymbol{L}. \tag{5.23}$$

For this reason,

$$\boldsymbol{n}ds = \boldsymbol{F}^{-T}J\boldsymbol{N}dS. \tag{5.24}$$

Surface integrals can then be expressed as:

$$\int_{\partial B^t} \boldsymbol{h}(\boldsymbol{x},t) \cdot \boldsymbol{n}(\boldsymbol{x})ds(\boldsymbol{x}) = \int_{\partial B^0} \boldsymbol{H}(\boldsymbol{X}) \cdot \boldsymbol{F}^{-T}(\boldsymbol{X},t)\boldsymbol{N}(\boldsymbol{X})J(\boldsymbol{X},t)dS(\boldsymbol{X}) \tag{5.25}$$

5.1.3. Governing Equations

Conservation of mass and conservation of momentum can be written both in Lagrangian view using the first Piola-Kirchoff stress P

$$\frac{\partial}{\partial t}(R(\boldsymbol{X},t)J(\boldsymbol{X},t)) = 0 \qquad Conservation\,of\,mass, \tag{5.26}$$

$$R(\boldsymbol{X},0)\frac{\partial \boldsymbol{V}}{\partial t} = \nabla^{\boldsymbol{X}} \cdot \boldsymbol{P} + R(\boldsymbol{X},0)\boldsymbol{g} \qquad Conservation\,of\,momentum, \tag{5.27}$$

and in Eulerian view adopting the Cauchy stress σ

$$\frac{D}{Dt}\rho(\boldsymbol{x},t) + \rho(\boldsymbol{x},t)\nabla^{\boldsymbol{x}} \cdot \boldsymbol{v}(\boldsymbol{x},t) = 0 \qquad Conservation\,of\,mass, \tag{5.28}$$

$$\rho(\boldsymbol{x},t)\frac{D\boldsymbol{v}}{Dt} = \nabla^{\boldsymbol{x}} \cdot \boldsymbol{\sigma} + \rho(\boldsymbol{x},t)\boldsymbol{g} \qquad Conservation\,of\,momentum, \tag{5.29}$$

considering the material derivative $\frac{D}{Dt} = \frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla^{\boldsymbol{x}}$

Conservation of Mass

Defining B_ϵ^t as a ball of radius ϵ surrounding an arbitrary $\boldsymbol{x} \in \Omega^t$, The density ρ is

$$\rho(\boldsymbol{x},t) = \lim_{\epsilon \to +0} \frac{mass(B_\epsilon^t)}{\int_{B_\epsilon^t} d\boldsymbol{x}} \tag{5.30}$$

Conservation of mass can be expressed as

$$mass(B_\epsilon^t) = \int_{B_\epsilon^t} \rho(\boldsymbol{x}, t) d\boldsymbol{x} = \int_{B_\epsilon^0} R(\boldsymbol{X}, t) J d\boldsymbol{X} = \int_{B_\epsilon^0} R(\boldsymbol{X}, 0) d\boldsymbol{X} = mass(B_\epsilon^0) \quad (5.31)$$

for all $B_\epsilon^t \subset \Omega^t$, with

$$R(\boldsymbol{X}, t)J(\boldsymbol{X}, t) = R(\boldsymbol{X}, 0), \qquad \forall \boldsymbol{X} \in \Omega^0, \qquad t \geq 0. \quad (5.32)$$

It is possible to write mass conservation as:

$$\frac{\partial}{\partial t}(R(\boldsymbol{X}, t)J(\boldsymbol{X}, t)) = 0. \quad (5.33)$$

Recalling

$$\frac{\partial}{\partial t}(RJ) = \frac{\partial R}{\partial t}J + R\frac{\partial J}{\partial t}, \quad (5.34)$$

and

$$\frac{\partial J}{\partial \boldsymbol{F}}(RJ) = J\boldsymbol{F}^{-T}, \quad (5.35)$$

$$\frac{\partial J}{\partial t} = \frac{\partial J}{\partial F_{ij}}\frac{\partial F_{ij}}{\partial t} = JF_{ij}^{-1}\frac{\partial V_i}{\partial X_j} = JF_{ij}^{-1}\frac{\partial v_i}{\partial x_k}F_{kj} = J\delta_{ik}\frac{\partial v_i}{\partial x_k} = J\frac{\partial v_i}{\partial x_i}. \quad (5.36)$$

Combining Equation 5.33, 5.34, 5.36, the obtained result is

$$\frac{\partial R}{\partial t}J + RJ\frac{\partial v_i}{\partial x_i} = 0. \quad (5.37)$$

The Eulerian perspective of the Equation 5.37 can be obtained by pushing forward both sides, resulting in:

$$\frac{D}{Dt}\rho(\boldsymbol{x}, t) + \rho(\boldsymbol{x}, t)\nabla^{\boldsymbol{x}} \cdot \boldsymbol{v}(\boldsymbol{x}, t) = 0, \qquad \forall \boldsymbol{x} \in \Omega^t, \qquad t \geq 0. \quad (5.38)$$

Conservation of Momentum

Continuum forces are distinguished in volume forces or surface forces. To define the latter ones, let's assume the existence of a traction field $\boldsymbol{t}(\cdot, \boldsymbol{n}, t) : \Omega^t \to \mathbb{R}^d$ defined as

$$forces(B_\epsilon^t) = \int_{\partial B_\epsilon^t} \boldsymbol{t}(\boldsymbol{x}, \boldsymbol{n}(\boldsymbol{x}))ds(\boldsymbol{x}) \tag{5.39}$$

where $forces(B_\epsilon^t)$ is the net force exchanged between two different sides of ∂B_ϵ^t

For this reason $\boldsymbol{t}(\boldsymbol{x}, \boldsymbol{n}, t)$ is the force per unit area exchanged, implying the presence of the Cauchy stress field $\sigma(\cdot, t) : \Omega^t \to \mathbb{R}^{dxd}$ with

$$\boldsymbol{t}(\boldsymbol{x}, \boldsymbol{n}, t) = \sigma(\boldsymbol{x}, t)\boldsymbol{n}. \tag{5.40}$$

The conservation of momentum is expressed as

$$\frac{d}{dt}\int_{B_\epsilon^t} \rho(\boldsymbol{x}, t)\boldsymbol{v}(\boldsymbol{x}, t)d\boldsymbol{x} = \frac{d}{dt}\int_{B_\epsilon^0} R(\boldsymbol{X}, t)\boldsymbol{V}(\boldsymbol{X}, t)Jd\boldsymbol{X} = \int_{B_\epsilon^0} R(\boldsymbol{X}, t)\boldsymbol{A}(\boldsymbol{X}, t)d\boldsymbol{X} \tag{5.41}$$

$$= \int_{\partial B_\epsilon^t} \sigma\boldsymbol{n}ds(\boldsymbol{x}) + \int_{B_\epsilon^t} \boldsymbol{f}^{ext}d\boldsymbol{x} \tag{5.42}$$

for all $B_\epsilon^t \subset \Omega^t$. The external body force per unit volume is denoted as $\boldsymbol{f}^{ext}$. According to [10] $\sigma(\boldsymbol{x}, t)$ must be symmetric to guarantee angular momentum conservation.

Rewriting the right part of Equation 5.42

$$\int_{B_\epsilon^0} R(\boldsymbol{X}, t)J(\boldsymbol{x}, t)\boldsymbol{A}(\boldsymbol{X}, t)d\boldsymbol{X} = \int_{\partial B_\epsilon^t} \sigma\boldsymbol{n}ds(\boldsymbol{x}) + \int_{B_\epsilon^t} \boldsymbol{f}^{ext}d\boldsymbol{x} \tag{5.43}$$

and pushing forward the Lagrangian terms of Equation 5.43 results in:

$$\int_{B_\epsilon^t} \rho(\boldsymbol{x}, t)\boldsymbol{a}(\boldsymbol{X}, t)d\boldsymbol{x} = \int_{\partial B_\epsilon^t} \sigma\boldsymbol{n}ds(\boldsymbol{x}) + \int_{B_\epsilon^t} \boldsymbol{f}^{ext}d\boldsymbol{x} = \int_{B_\epsilon^t} \nabla^{\boldsymbol{x}} \cdot \sigma d\boldsymbol{x} + \int_{B_\epsilon^t} \boldsymbol{f}^{ext}d\boldsymbol{x}, \tag{5.44}$$

or

$$\rho \boldsymbol{a} = \nabla^{\boldsymbol{x}} \cdot \sigma + \boldsymbol{f}^{ext}, \qquad \forall \boldsymbol{x} \in \Omega^t, \qquad t \geq 0. \tag{5.45}$$

Conversely, the Eulerian terms of Equation 5.43 can be pulled back

$$\int_{\partial B_\epsilon^t} \sigma(\boldsymbol{x}, t) \boldsymbol{n} ds(\boldsymbol{x}) = \int_{\partial B_\epsilon^0} J(\boldsymbol{X}, t) \sigma(\phi(\boldsymbol{X}, t), t) \boldsymbol{F}^{-T}(\boldsymbol{X}, t) \boldsymbol{N} ds(\boldsymbol{X}). \tag{5.46}$$

Recalling that the first Piola-Kirchoff stress and the Cauchy stress are related by $\boldsymbol{P} = J\sigma \boldsymbol{F}^{-T}$, it results in

$$\int_{\partial B_\epsilon^t} \sigma(\boldsymbol{x}, t) \boldsymbol{n} ds(\boldsymbol{x}) = \int_{\partial B_\epsilon^0} \boldsymbol{P}(\boldsymbol{X}, t) \boldsymbol{N} ds(\boldsymbol{X}) = \int_{B_\epsilon^0} \nabla^{\boldsymbol{x}} \cdot \boldsymbol{P}(\boldsymbol{X}, t) d\boldsymbol{X}, \tag{5.47}$$

leading to the Lagrangian form of conservation of momentum after the combination with Equation 5.43:

$$\int_{B_\epsilon^0} R(\boldsymbol{X}, 0) \boldsymbol{A}(\boldsymbol{X}, t) d\boldsymbol{X} = \int_{B_\epsilon^0} \nabla^{\boldsymbol{x}} \cdot \boldsymbol{P}(\boldsymbol{X}, t) d\boldsymbol{X} + \int_{B_\epsilon^0} \boldsymbol{F}^{ext} J(\boldsymbol{X}, t) d\boldsymbol{X} \tag{5.48}$$

or

$$R(\boldsymbol{X}, 0) \boldsymbol{A}(\boldsymbol{X}, t) = \nabla^{\boldsymbol{x}} \cdot \boldsymbol{P}(\boldsymbol{X}, t) + \boldsymbol{F}^{ext}(\boldsymbol{X}, t) J(\boldsymbol{X}, t), \qquad \forall \boldsymbol{X} \in \Omega^0, \qquad t \geq 0 \tag{5.49}$$

Weak Form

Following [33], it is possible to ignore the external force for simplicity. The Lagrangian perspective of the conservation of momentum is the starting point

$$R(\boldsymbol{X}, 0) \boldsymbol{A}(\boldsymbol{X}, t) = \nabla^{\boldsymbol{x}} \cdot \boldsymbol{P}(\boldsymbol{X}, t), \qquad \forall \boldsymbol{X}, t \tag{5.50}$$

So for an arbitrary function $\boldsymbol{Q}(\cdot, t) : \Omega^0 \to \mathbb{R}^d$, the dot product to Equation 5.50 can be computed and integrated over Ω^0 to generate the weak form:

$$\int_{\Omega^0} Q_i(\boldsymbol{X}, t) R(\boldsymbol{X}, 0) A_i(\boldsymbol{X}, t) d\boldsymbol{X} = \int_{\Omega^0} Q_i(\boldsymbol{X}, t) P_{ij,j}(\boldsymbol{X}, t) d\boldsymbol{X} \tag{5.51}$$

$$= \int_{\Omega^0} ((Q_i(\boldsymbol{X},t)P_{ij}(\boldsymbol{X},t))_{,j} - Q_{i,j}(\boldsymbol{X},t)P_{i,j}(\boldsymbol{X},t))d\boldsymbol{X} \tag{5.52}$$

$$= \int_{\partial\Omega^0} Q_i(\boldsymbol{X},t)P_{ij}(\boldsymbol{X},t)N_j(\boldsymbol{X},t)ds(\boldsymbol{X}) - \int_{\Omega^0} Q_{i,j}(\boldsymbol{X},t)P_{ij}(\boldsymbol{X},t)d\boldsymbol{X}. \tag{5.53}$$

$P_{ij}N_j$ is the boundary condition of the problem. $\boldsymbol{T}(\boldsymbol{X},t)$ is the boundary force per unit surface with $T_i = P_{ij}N_j$. The force balance implies that $\forall \boldsymbol{Q}(\cdot,t) : \Omega^0 \to \mathbb{R}^d$

$$\int_{\Omega^0} Q_i(\boldsymbol{X},t)R(\boldsymbol{X},0)A_i(\boldsymbol{X},t)d\boldsymbol{X} = \int_{\partial\Omega^0} Q_iT_ids(\boldsymbol{X}) - \int_{\Omega^0} Q_{i,j}P_{ij}d\boldsymbol{X}. \tag{5.54}$$

Stress derivatives will be discretized in the present configuration: for this reason, the stress can be pushed forward

$$Q_{i,j} = \frac{\partial Q_i}{\partial X_j} = \frac{\partial q_i}{\partial x_k}\frac{\partial x_k}{\partial X_j} = q_{i,j}Fkj. \tag{5.55}$$

Besides, with

$$\sigma_{ik} = \frac{1}{J}P_{ij}F_{kj}, \tag{5.56}$$

Equation 5.54 becomes

$$\int_{\Omega^t} q_i(\boldsymbol{x},t)\rho(\boldsymbol{x},t)a_i(\boldsymbol{X},t)d\boldsymbol{X} = \int_{\partial\Omega^t} q_it_ids(\boldsymbol{X}) - \int_{\Omega^t} q_{i,k}\sigma_{ik}d\boldsymbol{x}. \tag{5.57}$$

Equation 5.57 is the weak form of force balance in the Eulerian perspective, the basis of the MPM discretization on the grid.

5.1.4. Constitutive Model

As mentioned in [19], an artificial equation of state is adopted to describe the pressure. For fluids, the stress field is expressed as

$$\boldsymbol{\sigma}^f = 2\mu_N\dot{\boldsymbol{\epsilon}} + \lambda_N tr(\dot{\boldsymbol{\epsilon}})\boldsymbol{I} - \hat{p}\boldsymbol{I} \tag{5.58}$$

where $\hat{p}$ is the pressure, μ_N is the shear viscosity and λ_N is the bulk viscosity. Let's recall

the strain rate tensor, $\dot{\boldsymbol{\epsilon}} = \frac{1}{2}(\boldsymbol{L} + \boldsymbol{L}^T)$ where $\boldsymbol{L}$ represents the gradient velocity tensor. The superscript f labels the fluid stress, to differentiate it from the one related to the fluid-solid interaction.

To represent a Newtonian, Stokesian fluid, $\lambda_N = \frac{2\mu_N}{3}$ is introduced, transforming the stress field into:

$$\boldsymbol{\sigma}^f = 2\mu_N\dot{\boldsymbol{\epsilon}} + \frac{2\mu_N}{3}tr(\dot{\boldsymbol{\epsilon}})\boldsymbol{I} - \hat{p}\boldsymbol{I} = 2\mu_N\left[\dot{\boldsymbol{\epsilon}} - \frac{1}{3}tr(\dot{\boldsymbol{\epsilon}})\boldsymbol{I}\right] - \hat{p}\boldsymbol{I} \tag{5.59}$$

with

$$\hat{p} = k\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right] \tag{5.60}$$

where ρ_0 is the initial density, k is the bulk modulus and $\gamma = 7$ for water.

5.1.5. Eulerian Interpolating Functions

At the beginning of the time step, the background Eulerian framework is created: for simplicity, a fixed Cartesian grid is generated. During each time step of MPM, the Lagrangian mass and momentum of material particles is transposed to the Eulerian grid nodes. After the governing equations are solved on the framework, velocities are transferred back to particles for them to perform the advection step: the grid's purpose is finished and it is destroyed. Both transfers require interpolation functions $N_i(\boldsymbol{x})$, defined over grid nodes $\boldsymbol{i}$. In 3D, $\boldsymbol{i} = (i, j, k)$. When $N_i(\boldsymbol{x})$ is evaluated at a particle location $\boldsymbol{x}_p$, a more concise notation $N_i(\boldsymbol{x}_p) = w_{ip}$ is adopted, representing a weight factor proportional to the interaction's strength.

Recalling [33], "dyadic products of one-dimensional interpolation functions are used as our grid basis functions as in [57]"

$$N_{\boldsymbol{i}}(\boldsymbol{x}_p) = N(\frac{1}{h}(x_p - x_i))N(\frac{1}{h}(y_p - y_i))N(\frac{1}{h}(z_p - z_i)), \tag{5.61}$$

where $\boldsymbol{i} = (i, j, k)$ is the grid index, $\boldsymbol{x}_p = (x_p, y_p, z_p)$ is the evaluation position, h is the grid spacing, $\boldsymbol{x}_i = (x_i, y_i, z_i)$ is the grid node position. For more compact notation, $w_{ip} = N_i(\boldsymbol{x}_p)$ and $\nabla w_{ip} = \nabla N_i(\boldsymbol{x}_p)$ will be used. To prevent "cell-crossing instability", interpolation function needs C_1 continuity: it is possible to use either quadratic B splines or cubic B splines [57].

The cubic kernel is defined with

$$N(x) = \begin{cases} \frac{1}{2}|x|^3 - |x|^2 + \frac{2}{3} & 0 \leq |x| < 1 \\ \frac{1}{6}(2 - |x|)^3 & 1 \leq |x| < 2 \\ 0 & 2 \leq |x| \end{cases} \tag{5.62}$$

while the quadratic one is:

$$N(x) = \begin{cases} \frac{3}{4} - |x|^2 & 0 \leq |x| < \frac{1}{2} \\ \frac{1}{2}(\frac{3}{2} - |x|)^2 & \frac{1}{2} \leq |x| < \frac{3}{2} \\ 0 & \frac{3}{2} \leq |x| \end{cases} \tag{5.63}$$

Computing the gradient is done similarly by differentiating the one dimensional functions, where $N'(x)$ is the derivative of $N(x)$.

Eulerian/Lagrangian Mass

Assign to each point a subset ($B^0_{\Delta x,p} \subset \Omega^0$) of the entire material, defining the mass of the particle as

$$m_p^n = \int_{B^{t^n}_{\Delta x,p}} \rho(\boldsymbol{x}, t^n) d\boldsymbol{x}. \tag{5.64}$$

To conservatively interpolate particles' mass, momentum and speed to the Eulerian framework, denote the mass of the grid node $\boldsymbol{i}$ as

$$m_i = \sum_p m_p N_i(\boldsymbol{x}_p) \tag{5.65}$$

leading to

$$\sum_i m_i = \sum_i \sum_p m_p N_i(\boldsymbol{x}_p) = \sum_p m_p \sum_i N_i(\boldsymbol{x}_p) = \sum_p m_p \tag{5.66}$$

assuming the partition of unity on $N_{\boldsymbol{i}}$.

Eulerian/Lagrangian Momentum

Let's transmit particle momentum $m_p \boldsymbol{v}_p$ as

$$(m\boldsymbol{v})_i = \sum_p m_p \boldsymbol{v}_p N_i(\boldsymbol{x}_p) \tag{5.67}$$

resulting in

$$\sum_i (m\boldsymbol{v})_i = \sum_p m_p \boldsymbol{v}_p \tag{5.68}$$

by the same logic shown in 5.1.5. Denote the Eulerian velocity $\boldsymbol{v}_i$ as

$$\boldsymbol{v}_i = \frac{(m\boldsymbol{v})_i}{m_i}. \tag{5.69}$$

Eulerian to Lagrangian Transfer

Since Lagrangian particle mass is always constant, velocity is easily interpolated as

$$\boldsymbol{v}_p = \sum_i \boldsymbol{v}_i N_i(\boldsymbol{x}_p), \tag{5.70}$$

verifying the conservation of momentum

$$\sum_p m_p \boldsymbol{v}_p = \sum_p m_p \sum_i \boldsymbol{v}_i N_i(\boldsymbol{x}_p) = \sum_i \boldsymbol{v}_i \sum_p m_p N_i(\boldsymbol{x}_p) = \sum_i m_i \boldsymbol{v}_i. \tag{5.71}$$

5.1.6. Discretization

Discrete Time

Recall the weak form of the force balance equation (Equation 5.54 and 5.57)

$$\int_{\Omega^0} Q_i R_0 A_i d\boldsymbol{X} = \int_{\partial\Omega^{t^n}} q_i t_i ds(\boldsymbol{x}) - \int_{\Omega^{t^n}} q_{i,k}\sigma_{ik} d\boldsymbol{x} \tag{5.72}$$

for all $\boldsymbol{q}(\boldsymbol{x}, t^n)$ (or $\boldsymbol{Q}(\boldsymbol{X}, t^n)$), at the t^n step time. The Lagrangian acceleration $A_i(\boldsymbol{X}, t^n)$ is substituted with $\frac{1}{\Delta t}(\hat{V}_i^{n+1} - V_i^n)$. Pushing forward the left side Ω^0 to Ω^t results in

$$\frac{1}{\Delta t}\int_{\Omega^{t^n}} q_i(\boldsymbol{x},t^n)\rho(\boldsymbol{x},t^n)(\hat{v}_i^{n+1}(\boldsymbol{x})-v_i^n(\boldsymbol{x}))d\boldsymbol{x}$$

$$=\int_{\partial\Omega^{t^n}} q_i(\boldsymbol{x},t^n)t_i(\boldsymbol{x},t^n)ds(\boldsymbol{x})-\int_{\Omega^{t^n}} q_{i,k}(\boldsymbol{x},t^n)\sigma_{ik}(\boldsymbol{x},t^n)d\boldsymbol{x}. \tag{5.73}$$

with $\boldsymbol{v}^n : \Omega^{t^n} \to \mathbb{R}^d$, $\hat{\boldsymbol{v}}^{n+1} : \Omega^{t^n} \to \mathbb{R}^d$ (both defined for $\boldsymbol{x} \in \Omega^{t^n}$). For this reason, $\hat{v}_i^{n+1}(\boldsymbol{x}) = V_i(\phi^{-1}(\boldsymbol{x},t^n),t^{n+1})$ and $v_i^n(\boldsymbol{x}) = V_i(\phi^{-1}(\boldsymbol{x},t^n),t^n)$. Note that the notation $\hat{\boldsymbol{v}}^{n+1}$ is chosen instead of $\boldsymbol{v}^{n+1}$ since the latter one is defined only in the domain of the next time step Ω^{tn+1}, [27]. All the i and k subscripts in Equation 5.73 are component index for dimensions ($i = 1,2,3$ for 3D and $i = 1,2$ for 2D). In the discussion of discrete space, $\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}$ will stand for a grid node. In the future, to avoid confusion, α,β,γ will refer to the component index, rewriting Equation 5.73 as

$$\frac{1}{\Delta t}\int_{\Omega^{t^n}} q_\alpha(\boldsymbol{x},t^n)\rho(\boldsymbol{x},t^n)(v_\alpha^{n+1}(\boldsymbol{x})-v_\alpha^n(\boldsymbol{x}))d\boldsymbol{x}$$

$$=\int_{\partial\Omega^{t^n}} q_\alpha(\boldsymbol{x},t^n)t_\alpha(\boldsymbol{x},t^n)ds(\boldsymbol{x})-\int_{\Omega^{t^n}} q_{\alpha,\beta}(\boldsymbol{x},t^n)\sigma_{\alpha\beta}(\boldsymbol{x},t^n)d\boldsymbol{x}. \tag{5.74}$$

where $\alpha,\beta = 1,2,...d$. $q_{\boldsymbol{i}\alpha}$ means the α component if the vector $\boldsymbol{q}$ stored at node $\boldsymbol{i}$, while $q_\alpha(\boldsymbol{x},t)$ denotes the α component of the field $\boldsymbol{q}(\boldsymbol{x},t)$.

Discrete Space

Replace q_α, v_α^n and v_α^{n+1} with grid based interpolants as

$$q_\alpha(\boldsymbol{x},t^n)=\sum_i q_{\boldsymbol{i}\alpha}(t^n)N_{\boldsymbol{i}}(\boldsymbol{x}), \tag{5.75}$$

$$v_\alpha^n(\boldsymbol{x})=\sum_j v_{\boldsymbol{j}\alpha}^n N_{\boldsymbol{j}}(\boldsymbol{x}), \tag{5.76}$$

$$v_\alpha^{n+1}(\boldsymbol{x})=\sum_j v_{\boldsymbol{j}\alpha}^{n+1} N_{\boldsymbol{j}}(\boldsymbol{x}) \tag{5.77}$$

or

$$q_\alpha^n = q_{i\alpha}^n N_i, \qquad v_\alpha^n = v_{j\alpha}^n N_j, \qquad v_\alpha^{n+1} = v_{j\alpha}^{n+1} N_j \tag{5.78}$$

for short with implied summation on the recurrent index. Force balance (Equation 5.74) becomes

$$\frac{1}{\Delta t}\int_{\Omega^{t^n}} q_{i\alpha}^n N_i(\boldsymbol{x})\rho(\boldsymbol{x},t^n)(v_{j\alpha}^{n+1}N_j(\boldsymbol{x}))d\boldsymbol{x} - \frac{1}{\Delta t}\int_{\Omega^{t^n}} q_{i\alpha}^n N_i(\boldsymbol{x})\rho(\boldsymbol{x},t^n)(v_{j\alpha}^{n}N_j(\boldsymbol{x}))d\boldsymbol{x}$$

$$= \int_{\partial\Omega^{t^n}} q_{i\alpha}^n N_i(\boldsymbol{x}) t_\alpha(\boldsymbol{x},t^n) ds(\boldsymbol{x}) - \int_{\Omega^{t^n}} q_{i\alpha}^n N_{i,\beta}(\boldsymbol{x})\sigma_{\alpha\beta}(\boldsymbol{x},t^n)d\boldsymbol{x} \tag{5.79}$$

for all $q_{i\alpha}$. Another possible definition is

$$q_{i\alpha}^n \delta_{\alpha\beta}\frac{m_{ij}^n}{\Delta t} v_{j\beta}^{n+1} - q_{i\alpha}^n \delta_{\alpha\beta}\frac{m_{ij}^n}{\Delta t} v_{j\beta}^{n}$$

$$= \int_{\partial\Omega^{t^n}} q_{i\alpha}^n N_i t_\alpha ds(\boldsymbol{x}) - \int_{\Omega^{t^n}} q_{i\alpha}^n N_{i,\beta}\sigma_{\alpha\beta} d\boldsymbol{x} \tag{5.80}$$

with

$$m_{ij}^n = \int_{\Omega^{t^n}} N_i(\boldsymbol{x})\rho(\boldsymbol{x},t^n)N_j(\boldsymbol{x})d\boldsymbol{x} \tag{5.81}$$

representing the mass matrix. Pulling back to the material results in

$$m_{ij}^n = \int_{\Omega^0} N_i(\boldsymbol{x}(\boldsymbol{X}))R(\boldsymbol{X},0)N_j(\boldsymbol{x}(\boldsymbol{X}))d\boldsymbol{X} \approx \sum_p m_p N_i(\boldsymbol{x}_p)N_j(\boldsymbol{x}_p). \tag{5.82}$$

It is symmetric positive semi-definite because it can be written as $\boldsymbol{B}\boldsymbol{B}^T$ ($m_{ij} = \sum_p B_{ip}B_{jp}$) where $B_{ip} = \sqrt{m_p}N_{ip}$ so that $\boldsymbol{z}^T\boldsymbol{M}\boldsymbol{z} \geq 0$ for any $\boldsymbol{z}$. In practice, this mass matrix cannot be used because it may be singular. To solve this issue, the mass matrix is approximated with a diagonal and positive definite matrix. This procedure is called "mass lumping" and the terms will be computed to be consistent with the particle-grid transfers.

Equation 5.81 is true for every $q_{i\alpha}^n$. So choosing them as

$$q^n_{i\alpha} = \begin{cases} 1, & \alpha = \hat{\alpha} \quad and \quad \boldsymbol{i} = \hat{\boldsymbol{i}} \\ 0, & otherwise \end{cases}$$

then

$$\sum_j \frac{m_{\hat{i}\boldsymbol{j}}}{\Delta t}(v^{n+1}_{j\hat{\alpha}} - v^n_{j\hat{\alpha}}) = \int_{\partial\Omega^{t^n}} N_{\hat{\boldsymbol{i}}} t_{\hat{\alpha}} ds(\boldsymbol{x}) - \int_{\Omega^{t^n}} N_{\hat{\boldsymbol{i}},\beta}\sigma_{\hat{\alpha}\beta} d\boldsymbol{x}, \tag{5.83}$$

a discrete force balance equation for the $\hat{\boldsymbol{i}}, \hat{\alpha}$ DoF on the grid. As previously mentioned, a less accurate but acceptable mass lumping simplification could be achieved by substituting the rows in m_{ij} with the corresponding row sum, transforming it into a diagonal matrix. The row sums called $\hat{m}_i$ for row $\boldsymbol{i}$ are

$$\hat{m}_i = \sum_j \int_{\Omega^{t^n}} N_{\boldsymbol{i}}(\boldsymbol{x})\rho(\boldsymbol{x}, t^n) N_{\boldsymbol{j}}(\boldsymbol{x}) d\boldsymbol{x} = \int_{\Omega^{t^n}} N_{\boldsymbol{i}}(\boldsymbol{x})\rho(\boldsymbol{x}, t^n) \sum_j N_{\boldsymbol{j}}(\boldsymbol{x}) d\boldsymbol{x}$$

$$= \int_{\Omega^{t^n}} N_{\boldsymbol{i}}(\boldsymbol{x})\rho(\boldsymbol{x}, t^n)(\boldsymbol{x}) d\boldsymbol{x}. \tag{5.84}$$

As previously done in Equation 5.82, since $m_p \approx V^0_p R(\boldsymbol{X}_p, 0)$, it is possible to approximate the following relation as

$$\int_{\Omega^{t^n}} N_{\boldsymbol{i}}(\boldsymbol{x})\rho(\boldsymbol{x}, t^n)(\boldsymbol{x}) d\boldsymbol{x} = \int_{\Omega^0} N_{\boldsymbol{i}}(\boldsymbol{x}(\boldsymbol{X}))R(\boldsymbol{X}, 0) d\boldsymbol{X} \approx \sum_p N_{\boldsymbol{i}}(\boldsymbol{x}_p) m_p, \tag{5.85}$$

meaning that $\hat{m}_{\boldsymbol{i}}$ will be approximated as $m_{\boldsymbol{i}}$.

Substituting $m_{\boldsymbol{i}}\boldsymbol{v}^n_{\boldsymbol{i}}$ with the momentum $(m\boldsymbol{v})^n_{\boldsymbol{i}}$, the discretization becomes

$$\frac{((m\boldsymbol{v})^{n+1}_{i\alpha} - (m\boldsymbol{v})^n_{i\alpha})}{\Delta t} = \int_{\partial\Omega^{t^n}} N_{\boldsymbol{i}}(\boldsymbol{x}) t_\alpha(\boldsymbol{x}, t^n) ds(\boldsymbol{x}) - \int_{\Omega^{t^n}} N_{\boldsymbol{i},\beta}(\boldsymbol{x})\sigma_{\alpha\beta}(\boldsymbol{x}, t^n) d\boldsymbol{x}. \tag{5.86}$$

The left hand side represents the change in momentum, while the right hand side is approximately the force. Recalling [33], which assumes as available an estimation of the stress $\boldsymbol{\sigma}^n_p \approx \sigma(\boldsymbol{x}^n_p, t^n)$ at each Lagrangian particle $\boldsymbol{x}^n_p$, thus

$$\int_{\Omega^{t^n}} N_{\boldsymbol{i},\beta}(\boldsymbol{x})\sigma_{\alpha\beta}(\boldsymbol{x},t^n)d\boldsymbol{x} \approx \sum_p \sigma^n_{p\,\alpha\beta} N_{\boldsymbol{i},\beta}(\boldsymbol{x}^n_p)V^n_p \tag{5.87}$$

where V^n_p is the volume particle p occupies at time t^n.

MLS-MPM momentum term

Following [27], the discretization of Equation 5.74 is different in the MLS-MPM approach. The continuum domain $\Omega^{t^n}_p$ is partitioned as

$$\int_{\Omega^{t^n}} \rho(\boldsymbol{x},t^n)v^n_\alpha(\boldsymbol{x})q_\alpha(\boldsymbol{x},t^n)d\boldsymbol{x} = \sum_p \int_{\Omega^{t^n}_p} \rho(\boldsymbol{x},t^n)v^n_\alpha(\boldsymbol{x})q_\alpha(\boldsymbol{x},t^n)d\boldsymbol{x}. \tag{5.88}$$

In each integral over $\Omega^{t^n}_p$, $\boldsymbol{x}$ is near $\boldsymbol{x}^n_p$, so, an estimated solution with nodal data samples is adopted for continuous equations

$$v^n_\alpha(\boldsymbol{x}) = \sum_{\boldsymbol{j}} \boldsymbol{\Phi_j}(\boldsymbol{x})v^n_{j\alpha} \tag{5.89}$$

and

$$q_\alpha(\boldsymbol{x},t^n) = \sum_{\boldsymbol{i}} \boldsymbol{\Phi}_{\boldsymbol{i}}(\boldsymbol{x})q^n_{i\alpha} \tag{5.90}$$

with the MLS shape function defined in [27]

$$\boldsymbol{\Phi_i}(\boldsymbol{x}) = \xi_i(\boldsymbol{x}^n_p)\boldsymbol{P}^T(\boldsymbol{x}-\boldsymbol{x}^n_p)\boldsymbol{M}^{-1}(\boldsymbol{x}^n_p)\boldsymbol{P}(\boldsymbol{x}_i-\boldsymbol{x}^n_p). \tag{5.91}$$

Therefore:

$$\sum_p \int_{\Omega^{t^n}_p} \rho(\boldsymbol{x},t^n)v^n_\alpha(\boldsymbol{x})q_\alpha(\boldsymbol{x},t^n)d\boldsymbol{x} = \sum_{p,\boldsymbol{i},\boldsymbol{j}} q^n_{i\alpha}v^n_{j\alpha}m_{ij}, \tag{5.92}$$

where $m_{ij} = \int_{\Omega^{t^n}_p} \rho(\boldsymbol{x},t^n)\boldsymbol{\Phi_i}(\boldsymbol{x})\boldsymbol{\Phi_j}(\boldsymbol{x})d\boldsymbol{x}$ is the mass matrix. Mass lumping previously reported in 5.1.6 is performed similarly here

$$m^n_i = \sum_p \int_{\Omega^{t^n}_p} \rho(\boldsymbol{x},t^n)\boldsymbol{\Phi_i}(\boldsymbol{x})d\boldsymbol{x} \approx \sum_p m_p\Phi_{\boldsymbol{i}}(\boldsymbol{x}^n_p) = \sum_p m_p N_{\boldsymbol{i}}(\boldsymbol{x}^n_p).$$

The grid velocity evolves from $\boldsymbol{v}_i^n$ to $\hat{\boldsymbol{v}}_i^{n+1}$. In simple terms, t^{n+1} velocities can be estimated around t^n particle positions $\boldsymbol{x}_p^n$ using $\hat{v}_\alpha^{n+1}(\boldsymbol{x}) = \boldsymbol{P}^T(\boldsymbol{x}-\boldsymbol{x}_p^n)\boldsymbol{c}_{\hat{v}_\alpha^{n+1}}(\boldsymbol{x}_p^n)$. Here, the subscript $\boldsymbol{c}$ represents the reconstructed physical term.

MLS-MPM stress term

The stress term on the right hand side of Equation 5.74 reveals the major contribution of MLS-MPM. Note, es exposed in [27], no arbitrary traction at the boundary is considered due to the assumption of a zero Neumann boundary condition. The stress integral can be defined through the summation over single particle domains, as previously done for the momentum term:

$$\int_{\Omega^{t^n}} q_{\alpha,\beta}(\boldsymbol{x},t^n)\sigma_{\alpha\beta}(\boldsymbol{x},t^n)d\boldsymbol{x} = -\sum_p \int_{\Omega_p^{t^n}} q_{\alpha,\beta}(\boldsymbol{x},t^n)\sigma_{\alpha\beta}(\boldsymbol{x},t^n)d\boldsymbol{x} \tag{5.93}$$

As in Equation 5.90, $\boldsymbol{q}(\boldsymbol{x},t)$ is expressed from a finite-dimensional function space, allowing to convert Equation 5.93 into a system of equations The resulting system contains [28]. $N_g d$ equations are obtained for all the DoFs, considering N_g grid nodes and d problem dimensions. For any grid node $\hat{\boldsymbol{i}} \in 1,...,N_g$ and component $\hat{\alpha} \in 1,...,d$, for any degree of freedom, this choice of $\boldsymbol{q}$ is imposed in Equation 5.90 by setting

$$q_{\boldsymbol{i}\alpha}^n = \delta_{i\hat{i}}\delta_{\alpha\hat{\alpha}} \begin{cases} 1, & \alpha = \hat{\alpha} \quad and \quad \boldsymbol{i} = \hat{\boldsymbol{i}} \\ 0, & otherwise. \end{cases}$$

Recalling the MLS shape function reported in Equation 5.91 and combining it with this latter equation, it results in

$$q_\alpha(\boldsymbol{x},t^n) = \boldsymbol{P}^T(\boldsymbol{x}-\boldsymbol{x}_p^n)\boldsymbol{M}^{-1}(\boldsymbol{x}_p^n)\xi_{\hat{\boldsymbol{i}}}(\boldsymbol{x}_p^n)\boldsymbol{P}(\boldsymbol{x}_{\hat{i}}-\boldsymbol{x}_p^n)\delta_{\alpha\hat{\alpha}} \tag{5.94}$$

for any $\boldsymbol{x} \in \Omega_p^{t^n}$. Calculate this test functions over all $\hat{\boldsymbol{i}}$ and $\hat{\alpha}$ leads to the resulting force components $f_{\hat{\boldsymbol{i}}\hat{\alpha}}$ associated with all degrees of freedom on the framework.

Discretizing the Force

To reach the discrete force, Equation 5.93 requires the derivative of $\boldsymbol{q}$ with respect to $\boldsymbol{x}_\beta$, $q_{\alpha,\beta}$. Differentiating Equation 5.94 gives

$$q_{\alpha,\beta}(\boldsymbol{x}, t^n) = \frac{\partial \boldsymbol{P}^T(\boldsymbol{x} - \boldsymbol{x}_p^n)}{\partial x_\beta} \boldsymbol{M}^{-1}(\boldsymbol{x}_p^n) \xi_{\hat{\boldsymbol{i}}}(\boldsymbol{x}_p^n) \boldsymbol{P}(\boldsymbol{x}_{\hat{\boldsymbol{i}}} - \boldsymbol{x}_p^n) \delta_{\alpha\hat{\alpha}}. \tag{5.95}$$

The linear polynomial space $\boldsymbol{P}^T = [1, x_1 - x_{p1}^n, x_2 - x_{p2}^n, x_3 - x_{p3}^n]$ is adopted to facilitate the derivation, as well as choosing $\xi_{\hat{\boldsymbol{i}}} = N_{\hat{\boldsymbol{i}}}$ to be quadratic/cubic B-splines (to ensure that $\boldsymbol{M}^{-}1$ is a constant). With these assumptions, Equation 5.95 becomes

$$q_{\alpha\beta}(\boldsymbol{x}, t^n) = M_p^{-1} N_i(\boldsymbol{x}_p^n)(x_{\hat{\boldsymbol{i}}\beta} - x_{p\beta}) \delta_{\alpha\hat{\alpha}}, \tag{5.96}$$

where $M_p = \frac{1}{4}\Delta x^2$ for quadratic $N_i(\boldsymbol{x})$ and $\frac{1}{3}\Delta x^2$ for cubic $N_i(\boldsymbol{x})$.

Replacing it back into Equation 5.93 shows The $\hat{\alpha}$ component force computation on grid node $\hat{\boldsymbol{i}}$:

$$f_{\hat{\boldsymbol{i}}\hat{\alpha}} = -\sum_p \int_{\Omega_p^{t^n}} q_{\alpha,\beta}(\boldsymbol{x}, t^n) \sigma_{\alpha\beta}(\boldsymbol{x}, t^n) d\boldsymbol{x} \approx \sum_p V_p^n M_p^{-1} \sigma_{p_{\hat{\alpha}\beta}}^n N_{\hat{\boldsymbol{i}}}(\boldsymbol{x}_p^n)(x_{\hat{\boldsymbol{i}}\beta}^n - x_{p\beta}^n), \tag{5.97}$$

where V_p^n is the current volume of particle p at time n. To replace $\boldsymbol{\sigma}(\boldsymbol{x}, t^n)$ in Ω_p^n with $\boldsymbol{\sigma}_p^n$, a one-point quadrature rule approximation is used.

This explanation of the MLS-MPM algorithm mainly followed [27, 33]: further and more in-depth explanation regarding all the passages can be found in the mentioned articles.

5.2. Simulation Setup

5.2.1. Introduction of Water

As previously mentioned, the water is introduced inside the simulation with the usage of the Zibra Liquids plug-in, based on the MPM-MLS algorithm. A virtual tub of $3 \times 1.5 \times 0.25m^3$ is created. The grid resolution is set to 512 and about 5×10^6 particles of 5 mm are simulated in real time with the fixed time step value $\Delta t = 0.02s$, equal to the one of the `MonoBehaviour.FixedUpdate()` function of Unity. The dimensions of the tub, the grid and the particles are limited by the computational effort of simulating the fluid in real time. According to the Zibra Liquids Support Team, there is no possibility of slowing down the simulation and performing it in non-real time.
A liquid emitter is added to the container as well as the Articulation Body. In order to interact with the liquid, each object needs to have a Rigid Body component attached on it. However, this conflicts with the Articulation Body component already attached and needed to perform the inverse dynamics. The method used by the plug-in code to add forces and torques to the Rigid Body component is in common with the Articulation Body class: for this reason the Zibra Liquid Collider code was accessed and modified in order to allow the application of the force on each of the snake's modules. The mesh of the snake is sent to the Zibra Liquids server which generates the collider shape that interacts with the fluid particles.
As the simulation starts, the tub starts filling from the emitter. After some time, the velocity of the particles decreases and the generated currents disappear. To avoid the repetition of this initial step for every simulation, a first 60-minute simulation is performed once with the Articulation Body locked in its position and saved, so, each simulation can start from this "Baked Liquid" configuration, where the fluid is at rest.

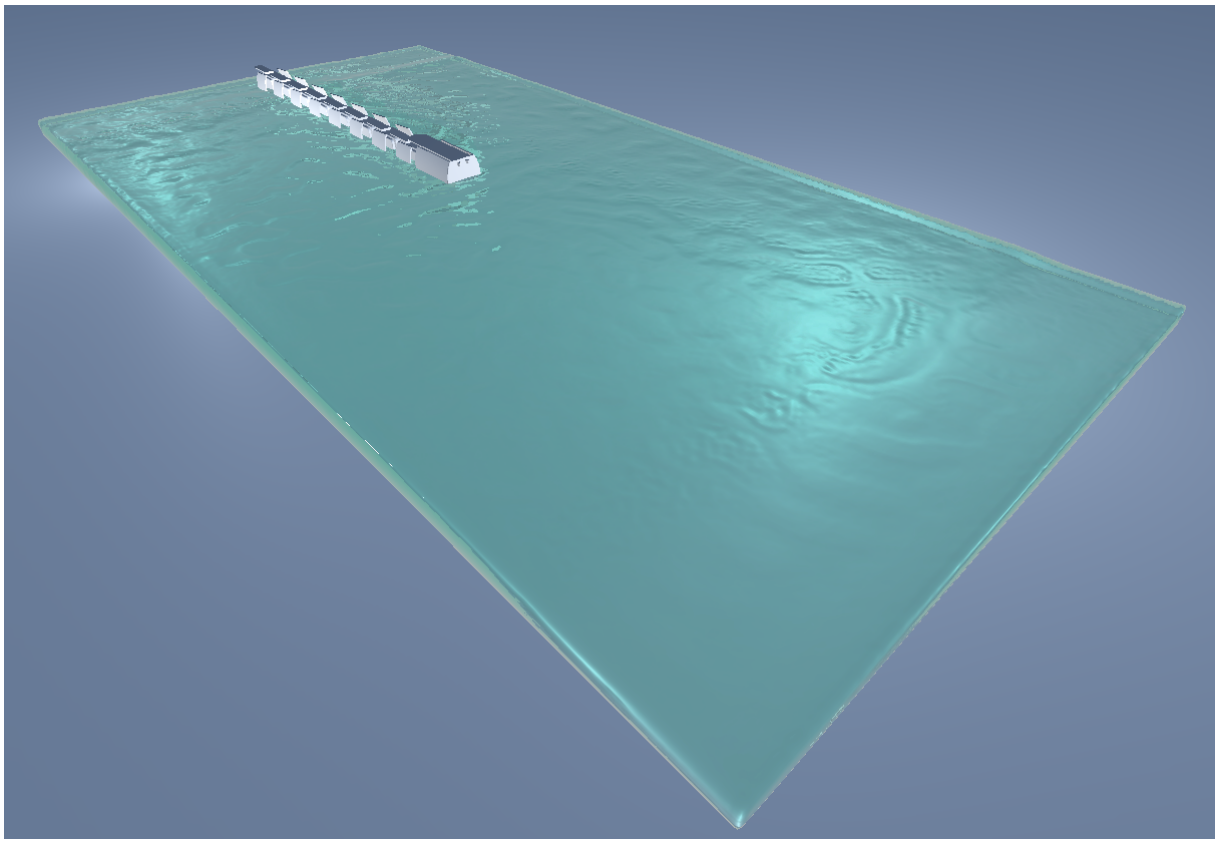
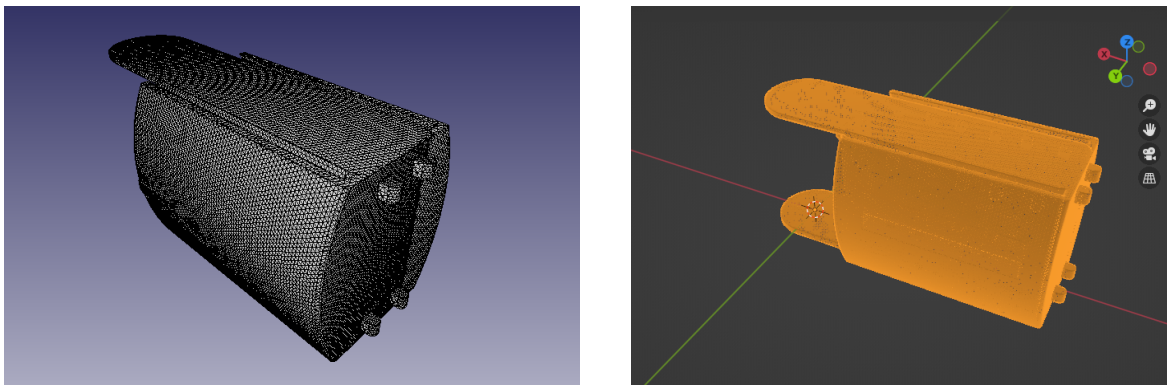


Figure 5.1: Zibra Liquids Virtual Tub.

5.2.2. Articulation Body creation

The CAD of all the modules constituting the snake robot was imported into Autodesk Inventor and re-oriented considering the different reference system of Unity. Each component constituting the assembly was weighted in laboratory and its mass and inertia properties were updated. The consequent mass, center of mass position and inertia tensor were calculated by Inventor and imported into Unity.
The assembly file of the head, the 7 modules and the last one have been simplified into a `.prt` file and exported as a `.stl` file. Using the Mephisto algorithm present in FreeCad, the meshes has been generated and exported as `.obj` files. In order to have consistent units into Unity, the meshes has been imported inside Blender, scaled down and re-exported as `.obj` file. Each of these files was then imported into Unity as a prefab.



(a) Generated mesh with Mephisto algorithm. (b) .obj file scaled down inside Blender.

Figure 5.2: The different steps needed to import the .obj file into Unity.

Each mesh was positioned into Unity, generating an Articulation Body. This component is crucial, since it allows modifying the parameters of mass, center of mass position and inertia tensor components. It is also possible to configure the single joint's DoF, introducing physical limits, joint drives with correspondent Force/Torques limitations and the target/target velocity to aim to. The snake robot Articulation Body was composed introducing +90°/-90° limits. In absence of a detailed servo motor's characteristic curve, the torque-speed dependency is assumed as linear, starting from the 1.52 Nm holding torque value to the 0 Nm associated to the maximum reachable speed of 6.54 rad/s, as shown in Figure 5.3.

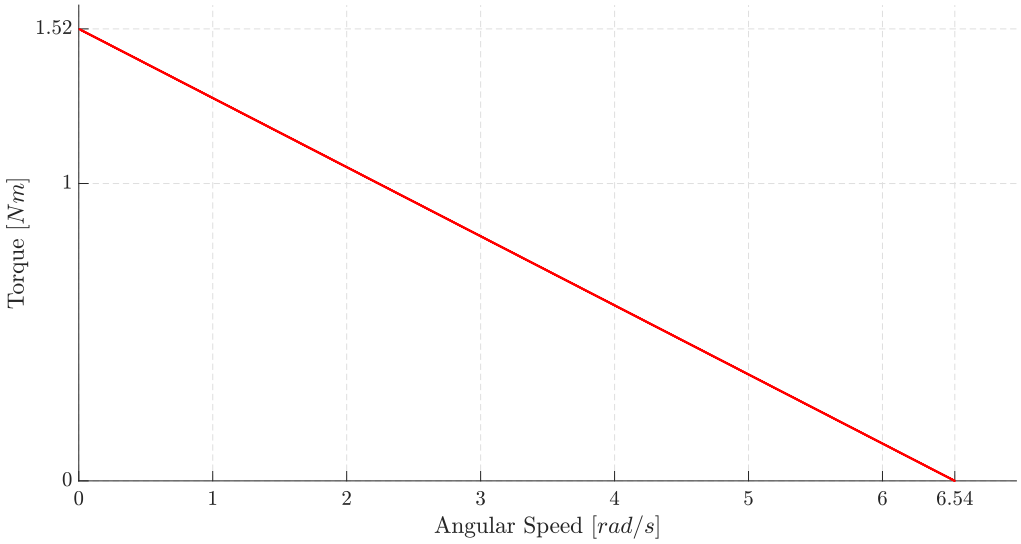


Figure 5.3: FeeTech FB5311M-360 characteristic curve.

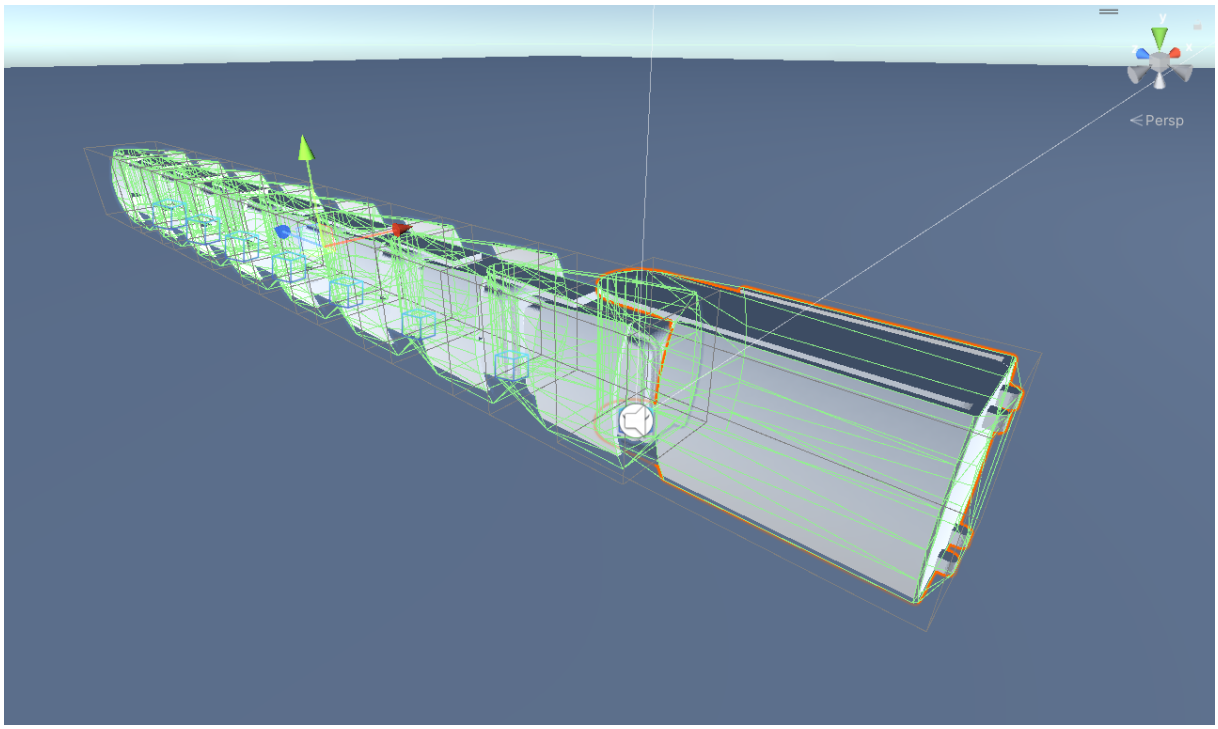


Figure 5.4: First robot articulation body inside Unity.

5.2.3. The coding of the simulation

Each articulation drive is able to apply a torque to reach a user-specified target. Two different $C\#$ codes were created, one to implement the eel-like motion law and the second one focused on the lateral-undulation motion law. The user is able to set amplitude A, frequency f, and phase shift φ for each motion law before the simulation starts. The duration of the initial transient λ can be modified inside the code. The targets are set with the function `ArticulationBody.SetDriveTargets()`, calculating for every $\Delta t = 0.02$ s, for each drive, the angular position to aim to.
The Lateral-Undulation motion target angles for each drive are coded as:

$$\theta_i = (1 - e^{-\lambda k \Delta t}) A \sin(\omega k \Delta t - i\varphi) \qquad i = 1, ..., 8. \tag{5.98}$$

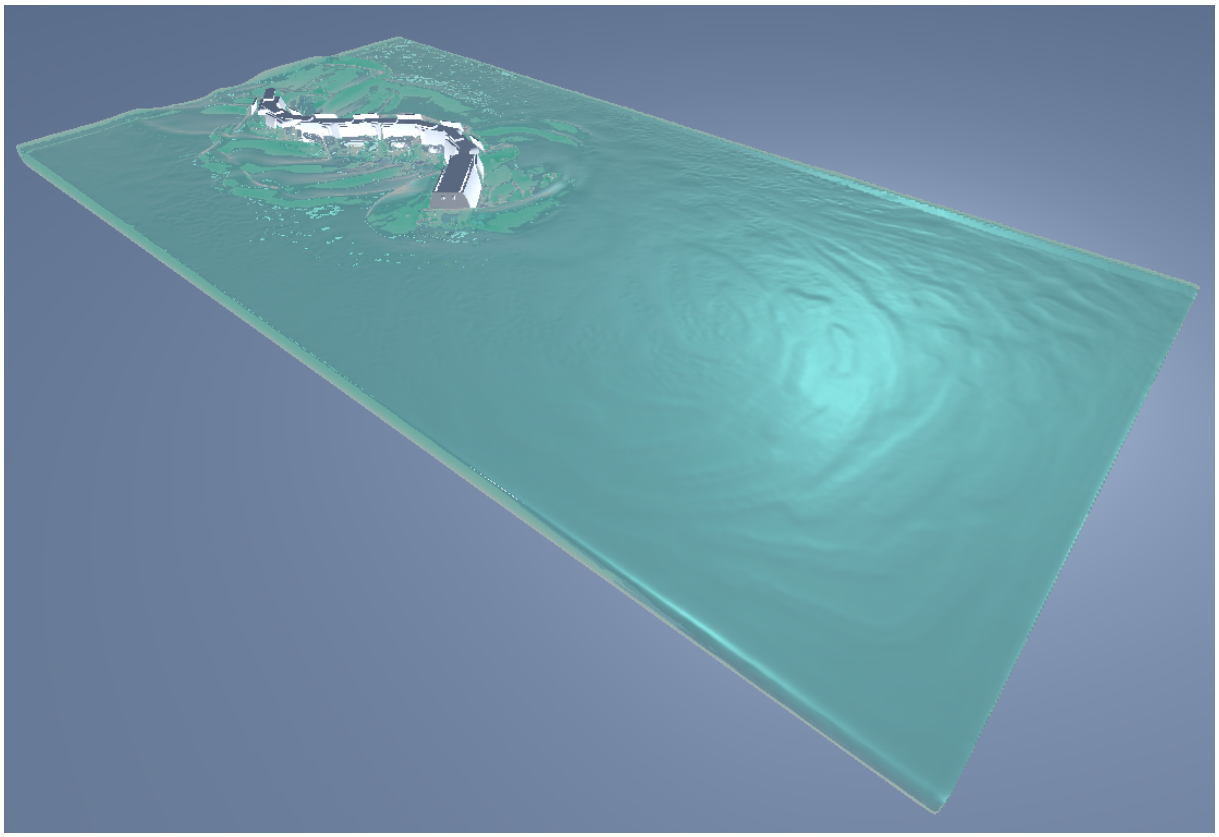


Figure 5.5: First robot performing lateral undulation motion.

The Eel-like motion target angles for each drive are coded as:

$$\theta_i = (1 - e^{-\lambda k \Delta t}) A \frac{i-1}{N+1} \sin(\omega k \Delta t - i\varphi), \qquad i = 1, ..., 8. \tag{5.99}$$

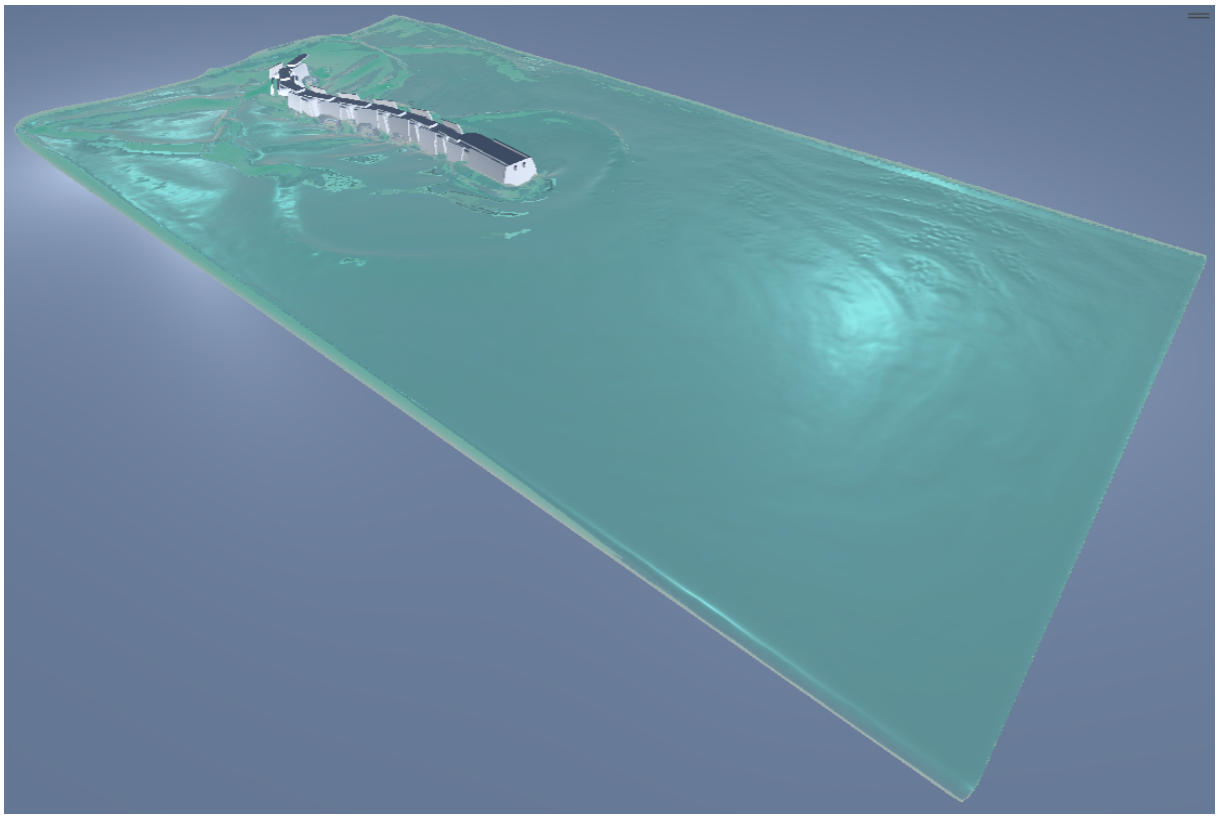


Figure 5.6: First robot performing eel-Like motion.

The simulation's duration is set to 10 s, while its time step is once again $\Delta t = 0.02$s: a vector of 500 values representing the time is created. Using the `MonoBehaviour.FixedUpdate()` function, each Articulation Body target has been accessed and modified, in order to follow the Lateral Undulation and Eel-like motion law. Before the beginning of the simulation, the snake's degrees of freedom are locked and, as the simulation starts, the brakes are removed. At the end of each `MonoBehaviour.FixedUpdate()`, the torques values are extracted from Unity with the command `ArticulationBody.GetDriveTorques()` that performs the inverse dynamics on the entire Articulation Body. The list of obtained values are stored in a string with the correspondent time instant and, at the end of the last `MonoBehaviour.FixedUpdate()`, the data are exported in a `.txt` file.

5.2.4. Inverse Dynamics

The inverse dynamics calculation is implemented inside Unity with the usage of the command `ArticulationBody.GetDriveTorques()`. The output of this command was compared to a MATLAB code that performs the inverse dynamics. The MATLAB code

considers a 2D problem, neglecting the vertical forces equilibrium. The data related to angles and accelerations were exported from a simulation performed into Unity without water. The reference system is chosen consistently with the default one used by Unity. As shown in Figure 5.7, the following equilibrium between the head and the first module holds. The same equilibrium can be applied for the subsequent modules.

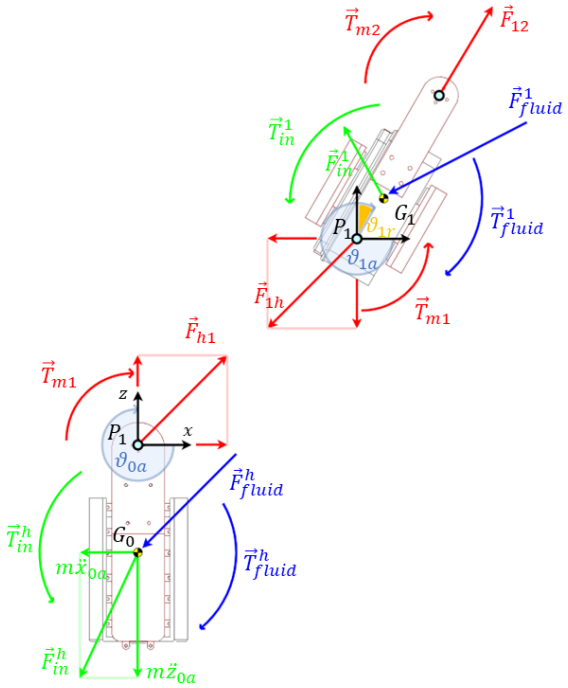


Figure 5.7: Equilibrium between the head and the first module.

$$\begin{cases} \boldsymbol{F^{h,1}} + \boldsymbol{F^h_{in}} + \boldsymbol{F^h_{fluid}} = \boldsymbol{0} & \text{(5.100a)} \\ \boldsymbol{T^1_m} + (\boldsymbol{G_0} - \boldsymbol{P_1}) \times \boldsymbol{F^h_{fluid}} + (\boldsymbol{G_0} - \boldsymbol{P_1}) \times \boldsymbol{F^h_{in}} + \boldsymbol{T^h_{fluid}} + \boldsymbol{T^h_{in}} = \boldsymbol{0} & \text{(5.100b)} \end{cases}$$

with

$$\boldsymbol{F^h_{in}} = -m_h \ddot{\boldsymbol{x}}^h_g,$$

$$\boldsymbol{T^h_{in}} = -J^h_G \ddot{\boldsymbol{\theta}}^h_g,$$

$$\ddot{\boldsymbol{x}}^h_g = \ddot{\boldsymbol{x}}_{\boldsymbol{O}} \ddot{\boldsymbol{\theta}}^h_a \boldsymbol{k} \times (\boldsymbol{G_h} - \boldsymbol{O}) - |\dot{\boldsymbol{\theta}}^h_a|^2 (\boldsymbol{G_h} - \boldsymbol{O})$$

with $\boldsymbol{F^{h,1}}$ is the force exchanged between the head and the first module, $\boldsymbol{F^h_{in}}$ is the inertial force of the head, $\boldsymbol{F^h_{fluid}}$ is the force that the liquid is applying to the head, $\boldsymbol{T^1_m}$ is the torque provided by the first drive, $(\boldsymbol{G_h} - \boldsymbol{P_1})$ is the vectorial distance between head's center of mass and the drive's vertical axis, $\boldsymbol{T^h_{fluid}}$ is the torque the plugin is generating on the center of mass of the head and $\boldsymbol{T^h_{in}}$ is the inertia torque of the head.

For the subsequent modules, the equilibrium equations are:

$$\begin{cases} -\boldsymbol{F}^{i-1,i} + \boldsymbol{F}^{i,i+1} + \boldsymbol{F}^{i}_{in} + \boldsymbol{F}^{i}_{fluid} = \boldsymbol{0} \\ -\boldsymbol{T}^{i}_{m} + \boldsymbol{T}^{i+1}_{m} + \boldsymbol{T}^{i}_{fluid} + \boldsymbol{T}^{i}_{in} + \\ +(\boldsymbol{G_i} - \boldsymbol{P_i}) \times \boldsymbol{F}^{i}_{fluid} + (\boldsymbol{G_i} - \boldsymbol{P_i}) \times \boldsymbol{F}^{i}_{in} + (\boldsymbol{P_{i+1}} - \boldsymbol{P_i}) \times \boldsymbol{F}^{i+1} = \boldsymbol{0} \end{cases}$$

with

$$\boldsymbol{F}^{i}_{in} = -m_i \ddot{\boldsymbol{x}}^{i}_{g},$$

$$\boldsymbol{T}^{h}_{in} = -J^{i}_{G} \ddot{\boldsymbol{\theta}}^{i}_{a},$$

$$\ddot{\boldsymbol{x}}^{i}_{p} = \ddot{\boldsymbol{x}}^{i-1}_{g} + \ddot{\boldsymbol{\theta}}^{i-1}_{a} \boldsymbol{k} \times (\boldsymbol{P_i} - \boldsymbol{G_{i-1}}) - |\dot{\boldsymbol{\theta}}^{h}_{a}|^2 (\boldsymbol{P_i} - \boldsymbol{G_{i-1}}),$$

$$\ddot{\boldsymbol{x}}^{i}_{g} = \ddot{\boldsymbol{x}}^{i}_{p} + \ddot{\boldsymbol{\theta}}^{i}_{a} \boldsymbol{k} \times (\boldsymbol{G_i} - \boldsymbol{P_i}) - |\dot{\boldsymbol{\theta}}^{i}_{a}|^2 (\boldsymbol{G_i} - \boldsymbol{P_i})$$

5.2.5. The Torque-estimator Algorithm

The combination of all the scripts running together with their execution order can be summarized in the following Algorithm 5.1

Algorithm 5.1 Torques Estimator

```
A, f, φ values set by the user
Start simulation
Save info string
while k < duration/dt do
    Set target angles θ
    Add torques T_m and reach the θ
    Update the fluid external forces F_f and torques T_f
    Get drive torques T_m applied to reach the targets θ
    Snake position updated due to the Physic Simulation
    if k < duration/dt then
        Update the string with t and T_m,
    end if
    k = k + 1
end while
Save torques data to file
Stop simulation
```

5.2.6. Simulation Results on the First Version of the Robot

The first robot was simulated as shown in Figures 5.5 and 5.6. According to the data extrapolated from the tests and shown in Figure 5.8, the first' robot servomotors were not able to provide the needed torque even without the application of fluid-dynamic forces.

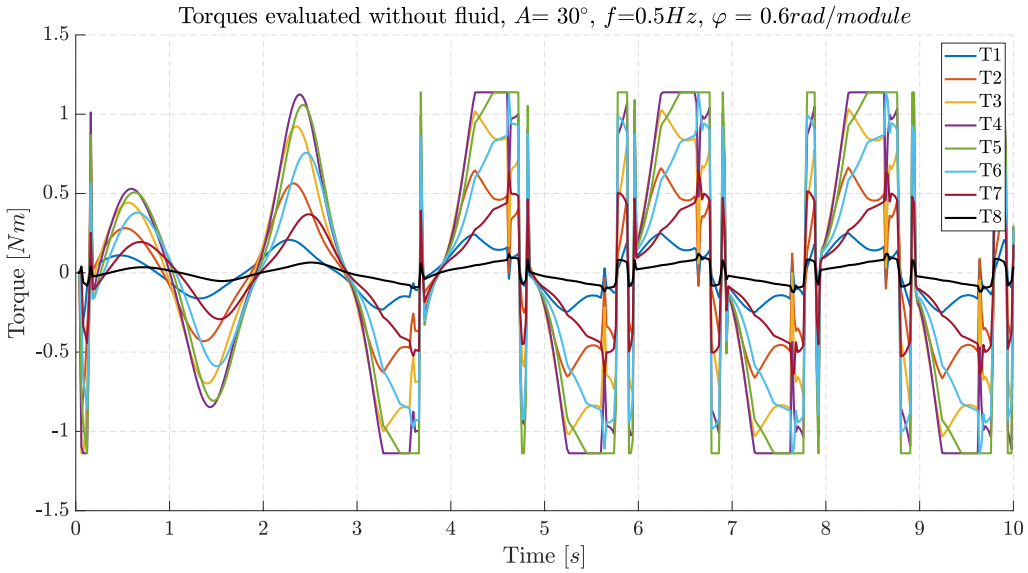


Figure 5.8: Inverse dynamics performed without the water application

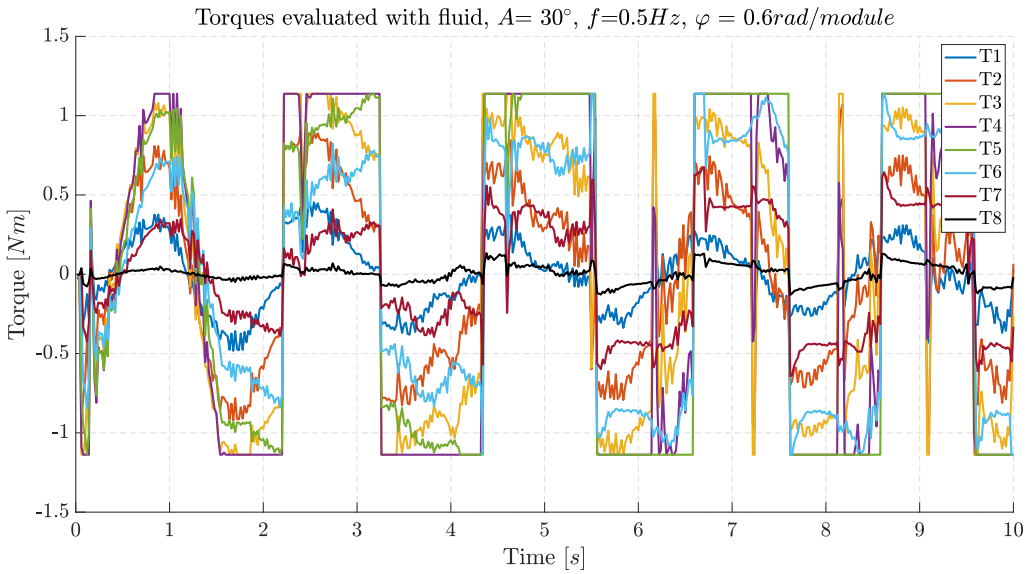


Figure 5.9: Inverse dynamics performed with the water application

Due to the previously mentioned limitations such as the circuit's fragility and the polyethy-

lene cover, since the updated motors would have needed severe modifications of the geometry, a complete re-design of the robot would allow to tackle the different weaknesses.

5.2.7. Second simulation

The new robot design with the correspondent inertial parameters and torques limitations was introduced inside the simulation. In absence of the actual torque-speed curve of the new servomotors, a cautionary descending linear behaviour starting from the maximum value of 3.43 Nm generated with a still motor supplied with 7.4 V and ending to a 0 Nm value at the maximum speed of 5.82 rad/s was considered.

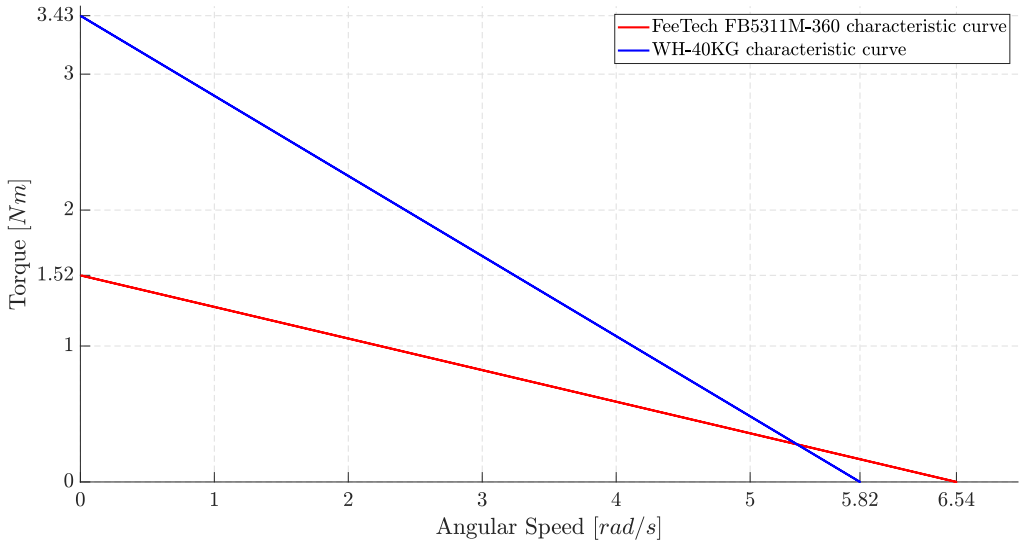


Figure 5.10: FeeTech FB5311M-360 and WH-40KG characteristic curves

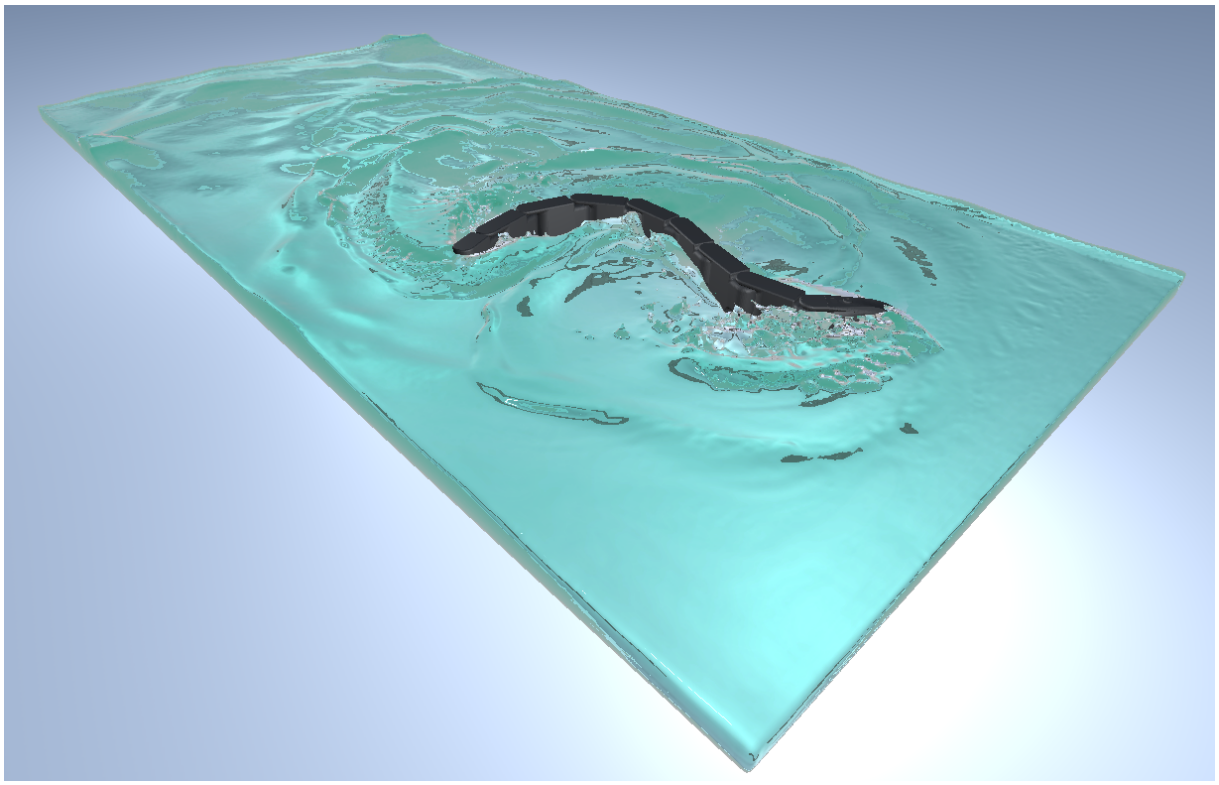


Figure 5.11: Second robot performing lateral undulation motion

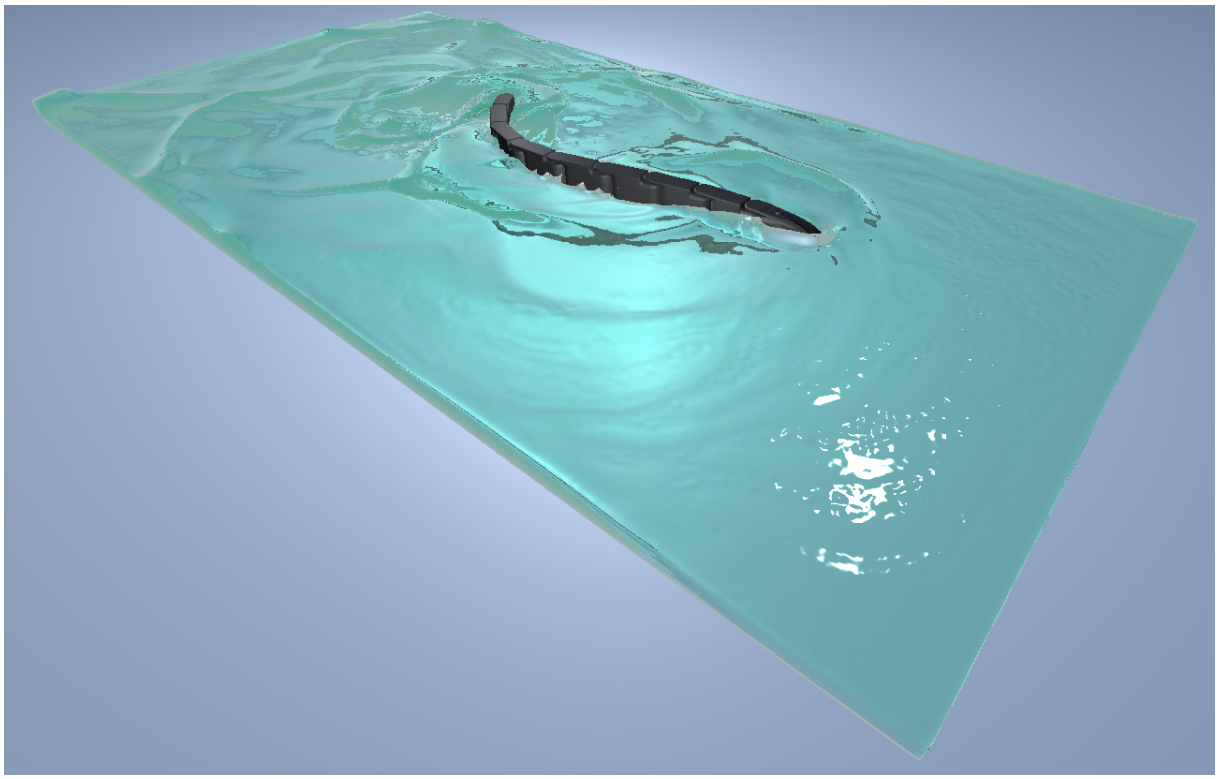


Figure 5.12: Second robot performing eel-like motion

For the same parameters tested for the first robot, the behaviour emerged is totally different, as reported in Figure 5.13 and Figure 5.14

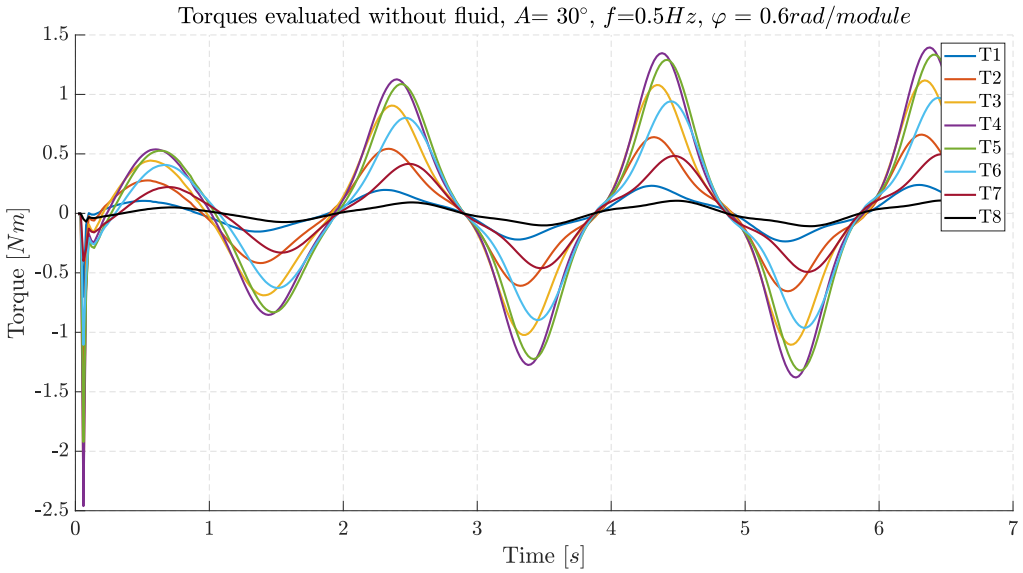


Figure 5.13: Inverse dynamics performed on the second robot without the water application

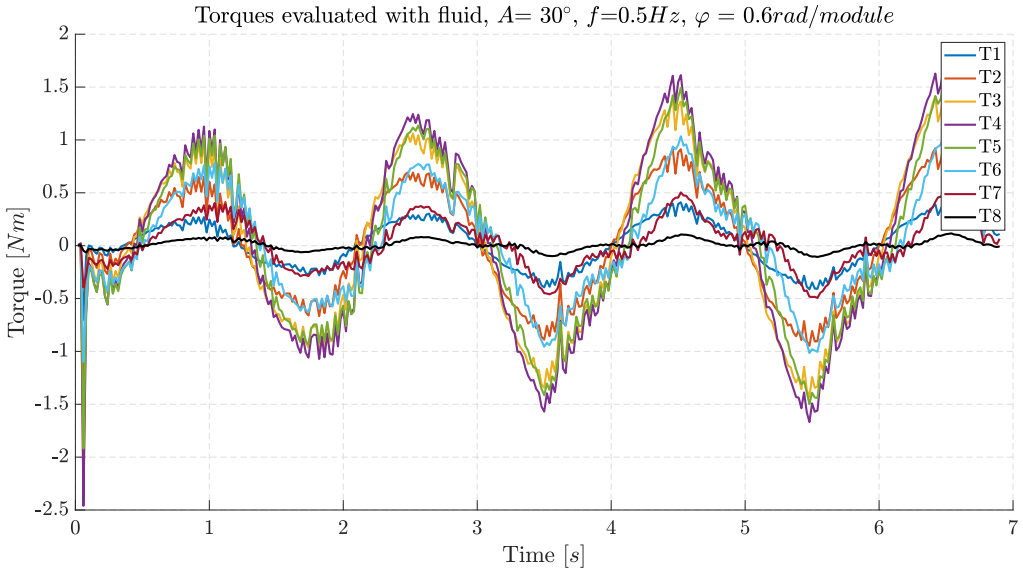


Figure 5.14: Inverse dynamics performed on the second robot with the water application

The aim of this simulation is to find an optimal region of possible motion's law parameters combination for the Lateral Undulation motion law. To do so, all the possible combinations have been tested, varying the amplitude parameters from 10° to 90°, the frequency

from 0.1Hz to 1.4 Hz and the phase shift among modules from 0.1 rad to 1.4 rad. To descend the different results, beyond the amplitude, frequency and phase parameters, the final value of the speed was stored as well as all the possible flags that would make the test null, such as the robot going backwards, self-colliding, saturating torque or non reasonable speed values in degenerated simulations. To speed up the testing process, an automatic loop process was implemented, that self triggers itself to automatically re-enter Unity's play mode with the new set of parameters to be tested.

The second simulation logic is summarized in the following Algorithm 5.2

Algorithm 5.2 Speed Estimator

1: A_{min}, A_{max}, A_{stp}, f_{min}, f_{max}, f_{stp}, φ_{min}, φ_{max}, φ_{stp} values set by the user, limit $\boldsymbol{Tm}$
2: Start simulation
3: Select Editor Update Loop window that triggers the play mode
4: Load data of previous A, f and φ
5: **if** $A == A_{max}$ **then**
6: **if** $f == f_{max}$ **then**
7: **if** $\varphi == \varphi_{max}$ **then**
8: Final stop = true
9: **else**
 $A = A_{min}$, $f = f_{min}$, $\phi = \phi + \varphi_{stp}$
10: **end if**
11: **else**
 $A = A_{min}$, $f = f + f_{stp}$
12: **end if**
13: **else**
 $A = A + A_{stp}$
14: **end if**
15: Store data of current A, f and φ
16: **if** $niter == 0$ **then**
17: Save speed infos on data file
18: **end if**
19: **while** $k < duration/dt$ **do**
20: Set target angles $\boldsymbol{\theta}$ with the current A, f and φ
21: Add torques $\boldsymbol{T_m}$ and reach the $\boldsymbol{\theta}$ and update the $\boldsymbol{F_f}$ and $\boldsymbol{T_f}$
22: Snake position updated due to the Physic Simulation
23: **if** $k == duration/dt$ **then**
24: Calculate v_m
25: Update the string with the current A, f, φ values, v_m and the corespondent invalid case
26: **end if**
27: k = k + 1
28: **end while**
29: Update speed data file and exit play mode
30: **if** Final stop == false **then**
31: Go to line 4
32: **else**
 Stop simulation
33: **end if**

5.2.8. Second Simulation Results

The second simulation's output is reported in Figure 5.15 and it consists of a 4D graph showing the obtained speed in correspondence of the tested parameters A, f and φ.

The range of parameters was chosen according to data shown in Table 5.1, for a total of 1764 simulations.

As emerged from the graph, not all the parameters combinations are feasible: some of them would cause the aquatic snake robot self-collide while others triads would need a torque level that the servo motor is not able to provide in that conditions. After having filtered out all the unfeasible combinations, a speed trend emerges, highlighting an optimal yellow zone characterized by a red dot representing the maximum achievable speed of 0.2955 m/s.

From the chart, some useful information can be extrapolated:

- Increasing values of φ lead to wider area where set of f and A give feasible results.
- Frequency and amplitude cannot be increased too much due to the limit in maximum torque of the motor.
- An optimal region was found, where the value of a maximum velocity of 0.2955 m/s, represented by the red dot, was found for $A = 30°$, $f = 0,9$Hz, $\varphi = 0,9$rad/module.

	Lower limit	**Upper limit**	**Unit**
A	10	90	deg
f	$0,1$	$1,4$	Hz
φ	$0,1$	$1,4$	rad

Table 5.1: List of parameters for simulation's test

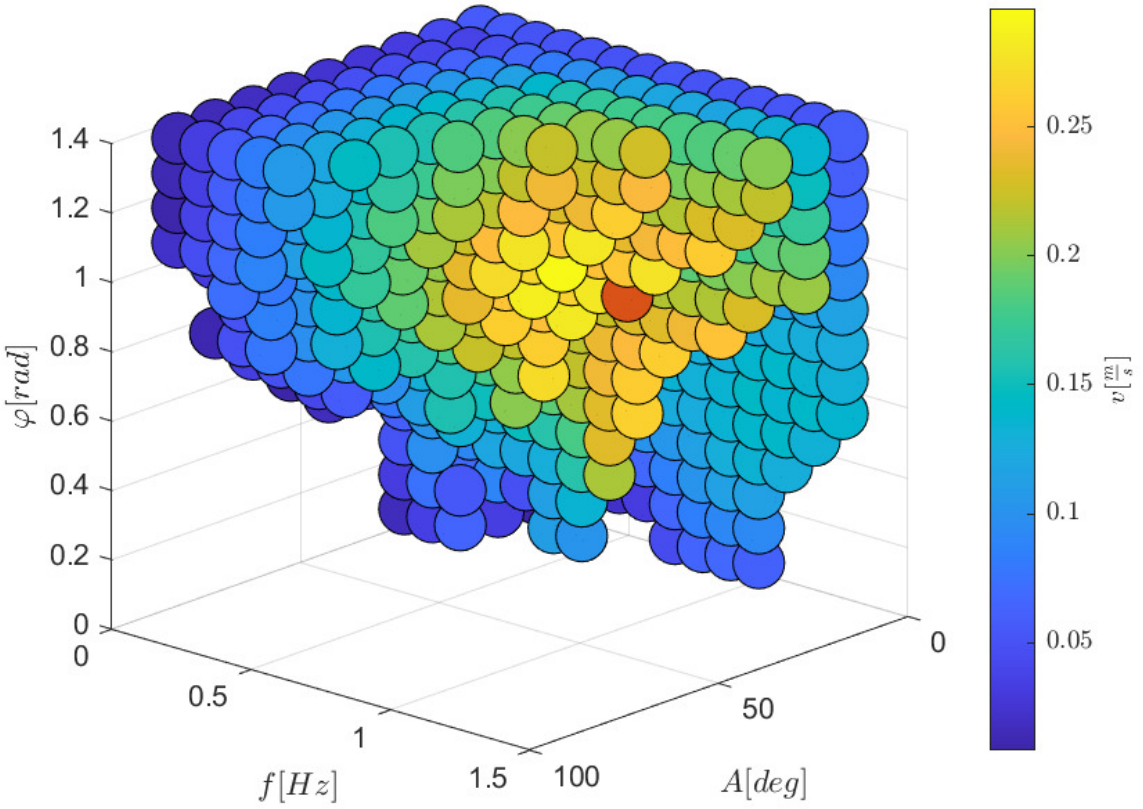


Figure 5.15: Simulation's result

6 | Control

In this chapter, the control strategy and its implementation are described. The main suddivision of the control strategy is:

- feedback on motor control,
- high-level control on motion law generation,

The goal is to create a robust feedforward control as well as to incorporate a feedback control for servomotors' target angles and obstacle avoidance.

6.1. Low level control system

The control at low level has two main tasks to be achieved:

- reach target angle,
- evaluation of target angle required based on motion law.

Target angles are reached by the use of two integrated PID controllers placed inside each module, while obstacle avoidance is obtained by integrating the system with an ultrasonic sensor together with magnetic encoders.

6.1.1. PID controller

Due to the presence of an already existent controller inside servomotors, target angle were reached very precisely even during first tests, however, in order to be able to assign specific values and do not rely just on an unknown component, we use an encoder, that gives us information about the real angle position. In this way, we are able to overtake the controller already present inside the servomotor if needed. Servomotors resulted to be very precise in reaching the target, as showed in Figure 6.1 target angle for the i^{th} module was reached without the addition of a supplementary closed-loop PID. However, the presence of encoder is crucial for obstacle avoidance task, and is going to be even more important during future work, were CPG algorithm or other control strategy may

be adopted.

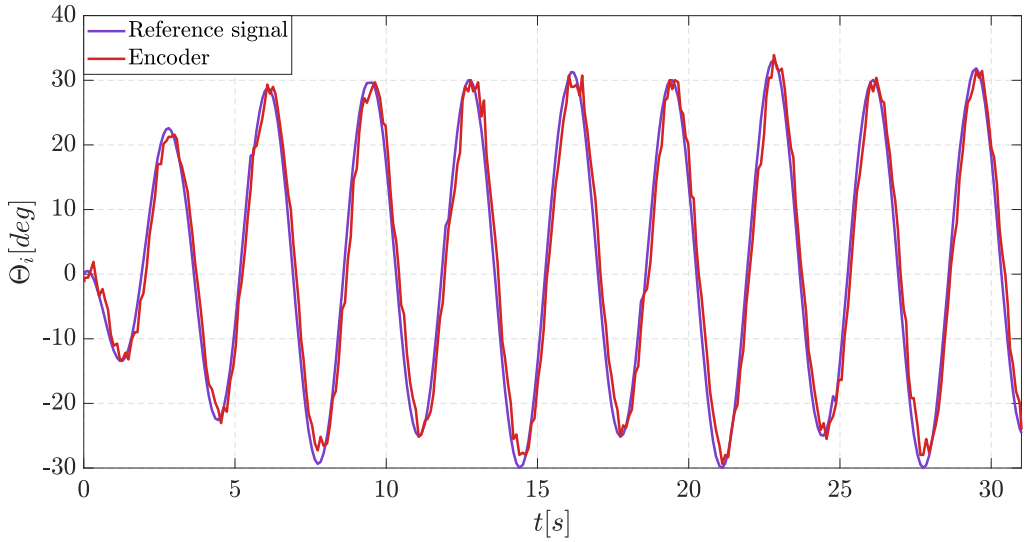


Figure 6.1: Test of encoders

6.2. Serpentine curve generation

Different strategies can be used for the generation of the serpentine curve [43]. This task is mainly divided into two parts, the generation of the momentum and the steering operation. During our work, we used two different strategies in order to make the robot move forward.

- lateral undulation,
- eel-like motion.

Lateral undulation is known in the literature to be the best solution in terms of reachable speed [43]. However, eel-like is interesting since provides good forward speed despite a smaller average amplitude of oscillations, resulting in a slower but more efficient solution [53].

6.2.1. Lateral undulation

This is the fastest and most common form of snake locomotion and is considered in the majority of previous research on snake robots [43]. Moreover, we consider lateral undulation to be the most relevant to motion in cluttered environments, which is generally the main motivation behind research on snake robots [53]. Lateral undulation is achieved by creating continuous body waves that are propagated backward from head to tail [53]. A

well-known and common approach for achieving lateral undulation is to control the snake robot according to the serpenoid curve proposed by Hirose [25]. In particular, Hirose proposed that lateral undulation is realized by controlling each joint of the snake robot according to a sinusoidal reference. This model is represented by the following equation

$$\theta_i = A \sin(\omega t - i\varphi) \tag{6.1}$$

where A is the amplitude, ω is the frequency, φ the phase shift. While i represent the number of the joint, having the robot $N-1$ joints where N is the number of modules. Given that all those quantities are known, we are able to calculate θ_i which is the target angle evaluated for every joint i. The results are shown in Figure 6.2.

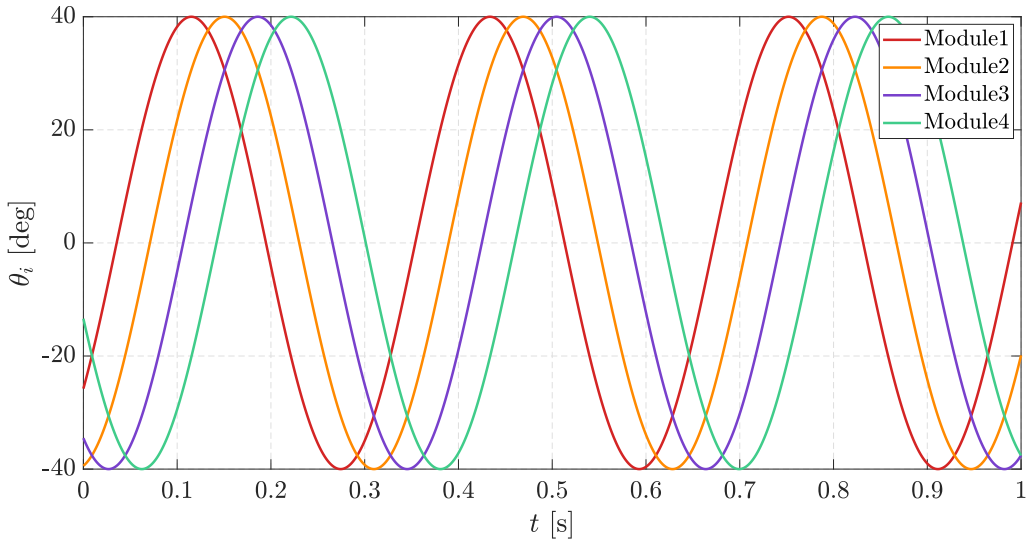


Figure 6.2: Reference signal for lateral undulation, 4 Modules, $f = 0.4$ Hz, $\varphi = 0.7$ rad, $A = 40°$

In order to limit sudden increase of torque during the initial phase of movement, the serpentine's curve equation is multiplied by an exponential function

$$exp = 1 - \mathrm{e}^{-\lambda t} \tag{6.2}$$

as follows,

$$\theta_i = (1 - \mathrm{e}^{-\lambda t}) A \sin(\omega t - i\varphi) \tag{6.3}$$

In this way, limited amplitude is achieved in initial stage of the movement, as shown in Figure 6.3.

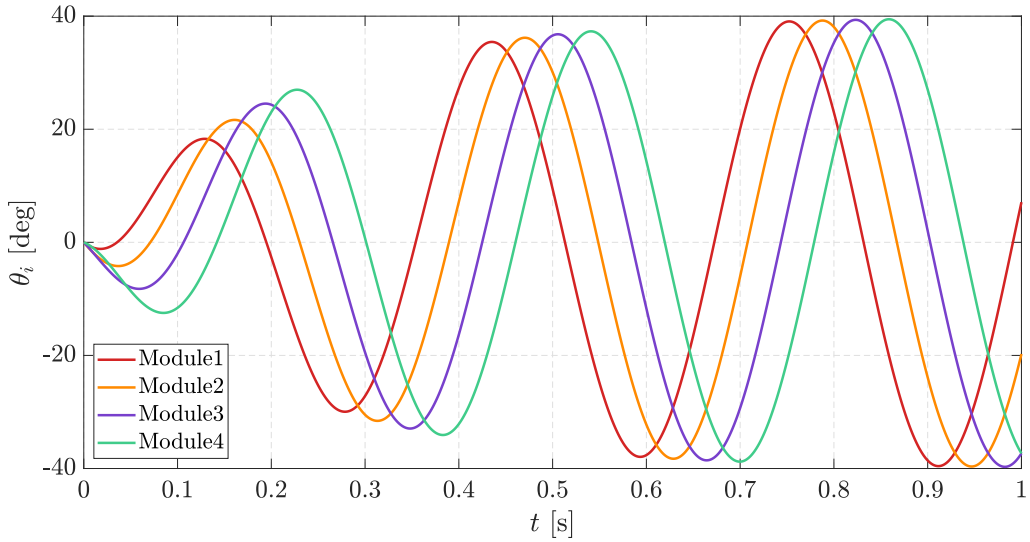


Figure 6.3: Reference signal for lateral undulation with the addition of transient, 4 Modules, $f = 0.4$Hz, $\varphi = 0.7$rad, $A = 40°$

6.2.2. Eel-like motion

Eel-like motion is another strategy that has a similar equation, with the addition of the factor $\frac{(i-1)}{(N+1)}$, where N represents the number of the modules. With this term the movement reaches a larger amplitude as the wave approaches the tail; thus, the tail performs large movement, and the head's rotation remains small so that it is always oriented in the same direction. Therefore, the final equation that we obtain for eel-like motion is

$$\theta_i = (1 - \mathrm{e}^{-\lambda t}) \frac{(i-1)}{(N+1)} A \sin(\omega t - i\varphi), \tag{6.4}$$

and is expression is described by the plot in Figure 6.4

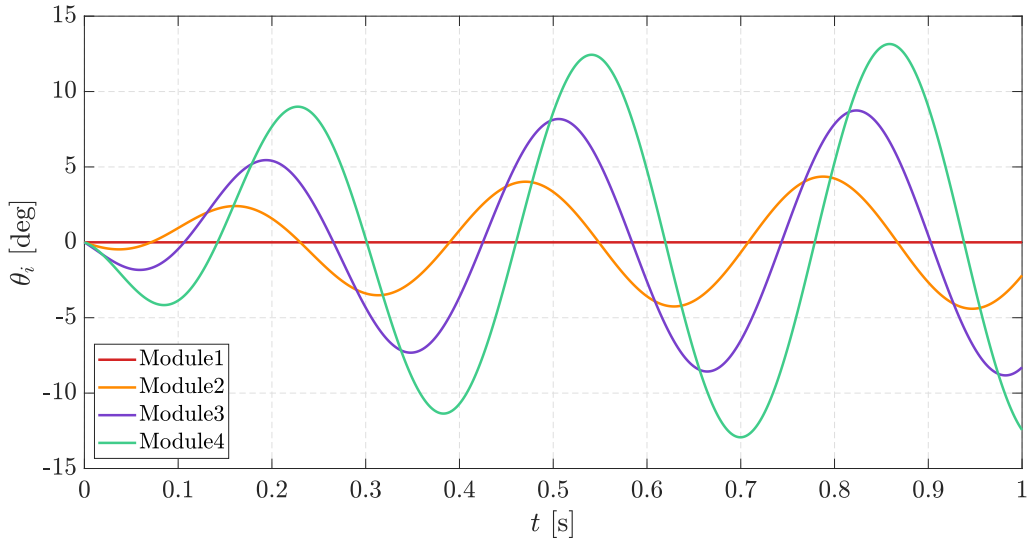


Figure 6.4: Reference signal for eel-like motion, 4 Modules, $f = 0.4$ Hz, $\varphi = 0.7$ rad, $A = 40°$

6.3. Steering

Another important aspect is the steering operation, and there are several ways to perform this task [45]. The two main solutions are implemented and tested to compare, in the experimental phase, which of the two is more effective. These solutions are:

- constant offset method
- constant hold method

6.3.1. Constant offset method

Consider, as stated before, the generation of the sinusoidal wave for the robot in its most general form:

$$\theta_i = g(i)A\sin(\omega t - i\varphi) \tag{6.5}$$

where $g(i)$ represent the type of motion:

- $g(i) = 1$ for lateral undulation
- $g(i) = \frac{(i-1)}{(N+1)}$ for eel-like motion

constant offset method consist of adding a constant phase offset φ_{offset} to the signal, so the obtained formulation is:

$$\theta_i = g(i)\sin(\omega t - i\varphi) + \varphi_{offset} \tag{6.6}$$

By passing this signal we perturb the straight motion of the robot and have it turn left or right depending on the sign of the phase offset, as presented in Figure 6.5. This solution is very simple and, indeed, does not reflect real sea snake behaviour [45].

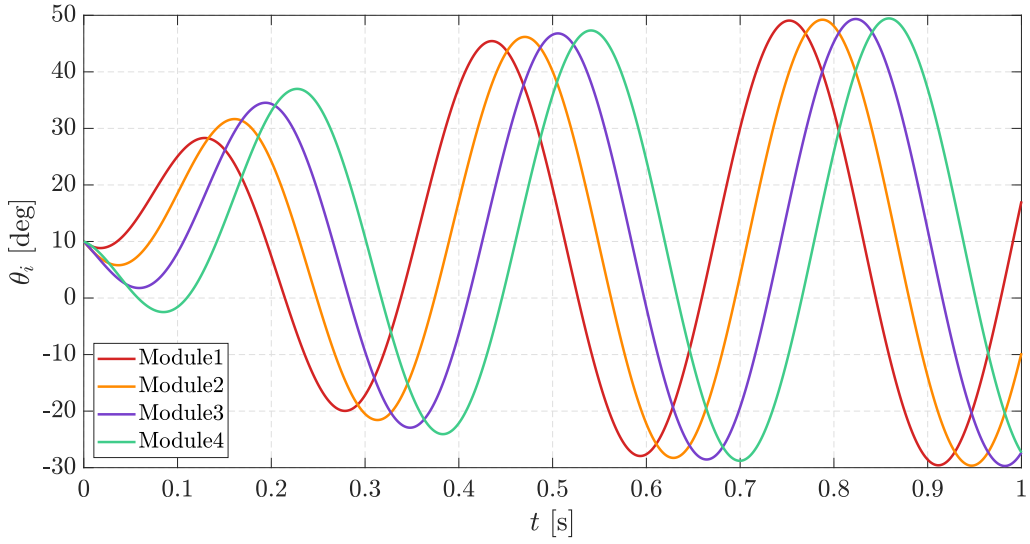


Figure 6.5: Reference signal for lateral undulation during steering in constant offset method, with an offset of 10°, 4 Modules, $f = 0.4$Hz, $\varphi = 0.7$rad, $A = 40°$

6.3.2. Constant-hold method

Constant-hold is a bioinspired strategy that aims to mimic real sea snake behavior during steering maneuvers [45]. To do so, it is needed to detect when the sinusoidal wave reaches a stationary point and hold the value for a fixed amount of time, doing this for a minimum or a maximum angle is reflected on the direction of steering. Even though this approach appears simple, it is less computationally efficient, and adding too complex control algorithms leads to heavy computations, which affects robot performance for generating the sinusoidal wave. To overcome this problem, we used a similar approach that is simpler from a computational point of view. In our solution, the sinusoidal wave is saturated before reaching his maximum value, as shown in Figure 6.6. In this way, we obtained a good behaviour both in terms of steering and reference signal's following.

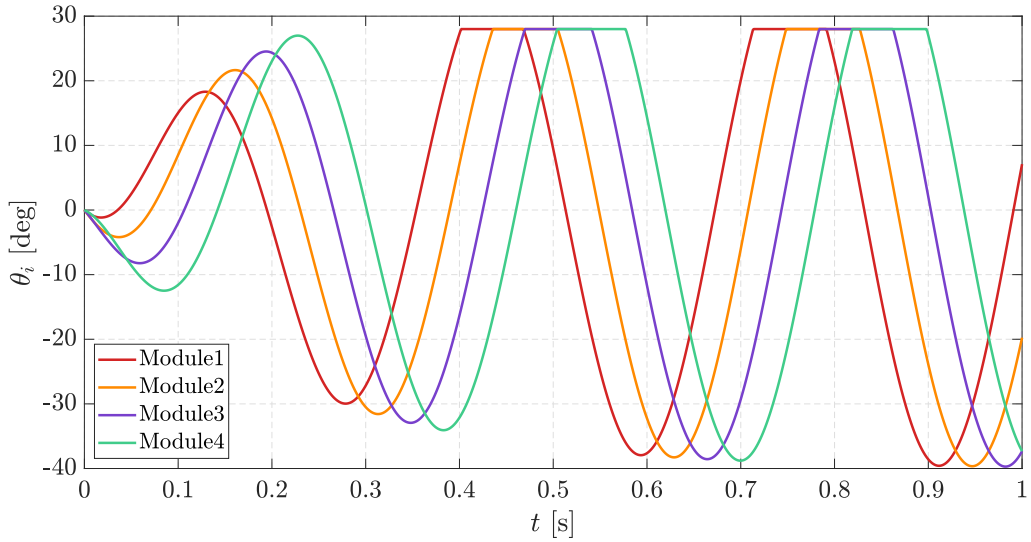


Figure 6.6: Reference signal of lateral undulation during constant-hold method, cut of amplitude at 70%, 4 Modules, $f = 0.4$Hz, $\varphi = 0.7$rad, $A = 40°$

6.4. High level control system

In order to make our robot follow the reference signal discussed above we implemented two controllers, one at high level and one at low level. The high-level controller consists of a feedforward action that starts from the users' device, the signal is then received from the Arduino Mega placed in the head through a bluetooth module. The overall architecture is shown in Figure 6.7: the board receives the command from an external user and then processes the evaluation of each target angle, which is then sent to the Arduino NANOs by using I2C protocol. In doing this, modules are task oriented, as they interpretate signals received from the Mega and they do not perform calculations regarding the generation of the serpentine curve.

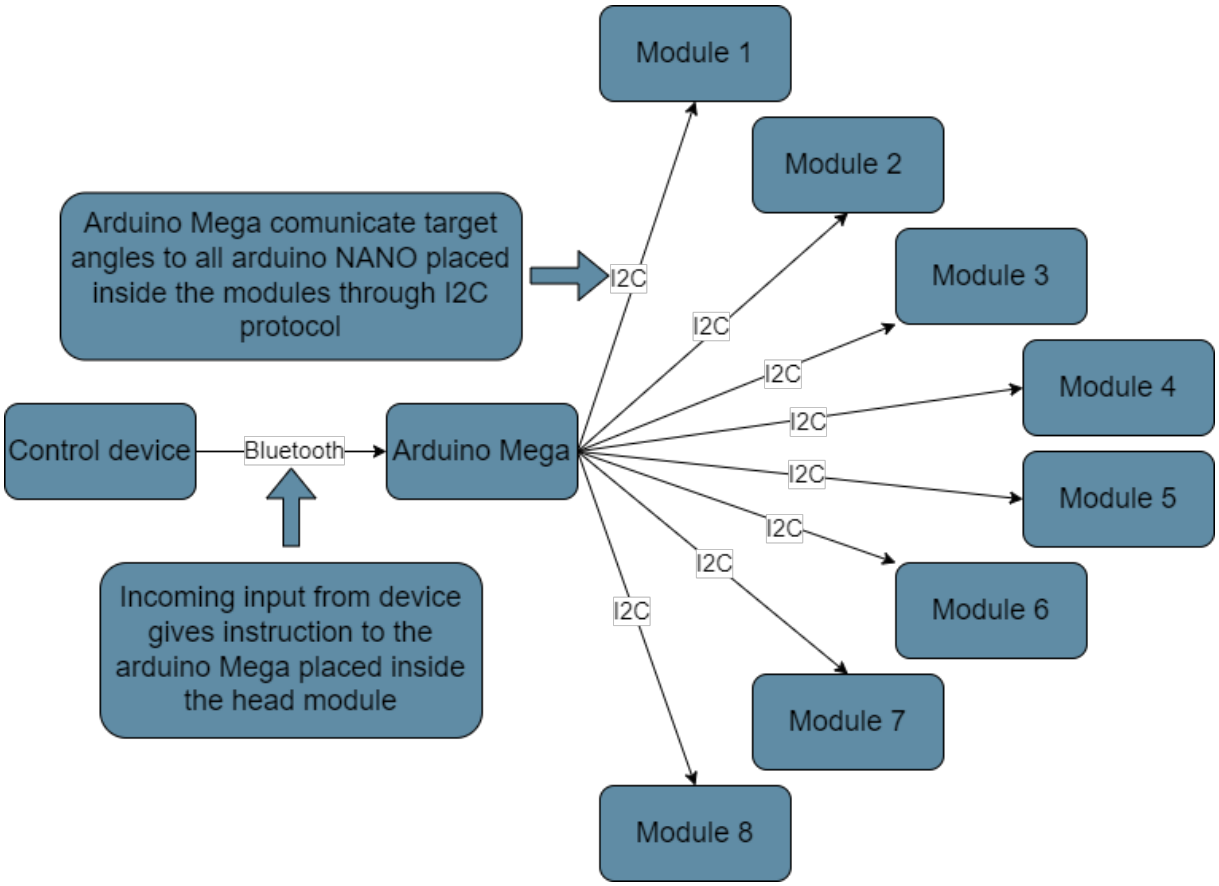


Figure 6.7: Block diagram of implementation

6.4.1. Obstacle avoidance

Due to the absence of GPS or external position triangulation systems, trajectory planning was not easy and was severely limited by the hardware at our disposal, making even simple task difficult to realize. For this reason the obstacle avoidance cannot be obtained through a position approach, and no optimal path can be found at the moment. In order to solve this problem for our purposes we decided to implement this function as a modification of the offset of the reference signal, as seen for the constant offset method with equation 6.6.

To correctly overcome obstacles two primal information are needed, measure of obstacle distance and orientation with respect to the main trajectory. First information is obtained through the ultrasonic sensor, in order to have correct estimation of the distance we had to be careful to the measure noise at low distance. For this reason, we integrated our program with the option of selecting a proper range of interest. Encoders were used to identify obstacle shift with respect to the main trajectory, which is a straight line, with the aim to adjust the shift value with the correct sign. This said, the reference signal for

the obstacle avoidance is identified by the following equation

$$\theta_i = g(i)A\sin(\omega t - i\varphi) + \varphi_{obstacle} \tag{6.7}$$

where $\varphi_{obstacle}$ is

$$\varphi_{obstacle} = -(\frac{\theta_{encoder}}{|\theta_{encoder}|})\frac{d}{D}K \tag{6.8}$$

where $\theta_{encoder}$ is the angle given by the encoder placed in the head of the robot, $\frac{d}{D}$ is the ratio between the distance of the obstacle detected by the sensor and the maximum distance that can be measured and K is a gain to increase the effect of control action. It is important to notice that summing $\varphi_{obstacle}$ at every time instant is not possible because continuous variations of offset severely affect the original reference signal leading to completely inefficient motion. For this reason, after the evaluation of the shift needed, $\varphi_{obstacle}$ is summed to the reference signal as a constant value for a fixed amount of time, optimal value for time was then be found with experiments, in order to have best combination both in terms of response's speed and vibration's bounding. However, also this solution appeared to be dangerous, for this reason two transients were introduced, in order to completely erase any sudden variation, therefore $\varphi_{obstacle}$ was evaluated as follows

$$\varphi_{obstacle}(t) = -(1 - \mathrm{e}^{-\lambda t})(\frac{\theta_{encoder}}{|\theta_{encoder}|})\frac{d}{D}K), \qquad t \in [0, \frac{T}{2}) \tag{6.9}$$

$$\varphi_{obstacle}(t) = -\mathrm{e}^{-\lambda(t-\frac{T}{2})}(\frac{\theta_{encoder}}{|\theta_{encoder}|})\frac{d}{D}K), \qquad t \in [\frac{T}{2}, T] \tag{6.10}$$

where T is the above discussed fixed amount of time. Results are shown in Figure 6.8.

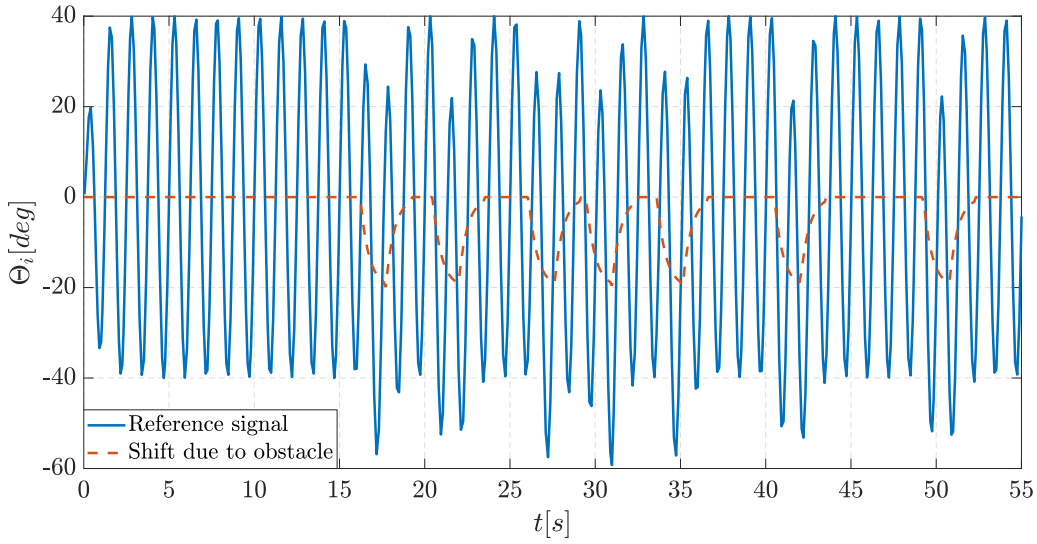


Figure 6.8: Reference signal compared with $\varphi_{obstacle}(t)$ ones the two transient are applied, $T = 3$s, $K = 20°$, obstacle positioned to the left side of main trajectory

7 | Experimental test

In this chapter, we will present the results obtained during the initial experimental tests. Our approach to these tests initially focused on verifying the correct execution of the tasks, both from the perspective of the control described in the previous chapter and with regard to the Arduino code used to implement them. Subsequently, our attention shifted towards validating the results obtained through simulations, aiming to confirm the correspondence between the velocities achieved based on the parameters used. Through a synergistic approach between these aspects, we endeavored to identify the optimal parameters to maximize speed, considering factors such as amplitude, frequency, and phase.

7.1. Task execution

The robot has four different tasks to be verified:

- lateral undulation
- eel-like motion
- steering operation in constant offset and constant hold
- obstacle avoidance

We are going to present last tree operations in lateral undulation only, in order to avoid redundancy of similar data.

7.1.1. Lateral undulation

At first we noticed that the robot does not sink, as expected, but while it is not swimming it tends to rotate and lay on its flat lateral surface. This problem does not influence the behavior, because as soon as the robot starts swimming it rapidly turns into a vertical position again. However, this unbalancing gives a small shift to the robot while it is swimming, leading it to constantly steer to the right. This problem was solved simply by adding a constant positive shift of 3° to the reference signal. In this way, the robot was able to swim following a rectilinear path. Actually, having the robot continuously steering

is useful from an experimental point of view, the closer the distance between the robot and the operator, the fewer disconnections and loss of time due to repositioning. For those reasons, this offset is kept while testing the tasks, and it is adjusted only when the measure of speed, as well as showing the steering feature, is needed. Lateral undulation was then tested with parameters shown in Table 7.1, for which a speed of $21\frac{cm}{s}$ was obtained. Data relative to the reference signal for each module are shown in Figure 7.1 while frame of the experiment's video are shown in Figure 7.2

	Value	Unit
A	20	deg
f	$0,5$	Hz
φ	$\frac{\pi}{6}$	rad

Table 7.1: List of parameters for lateral undulation's test

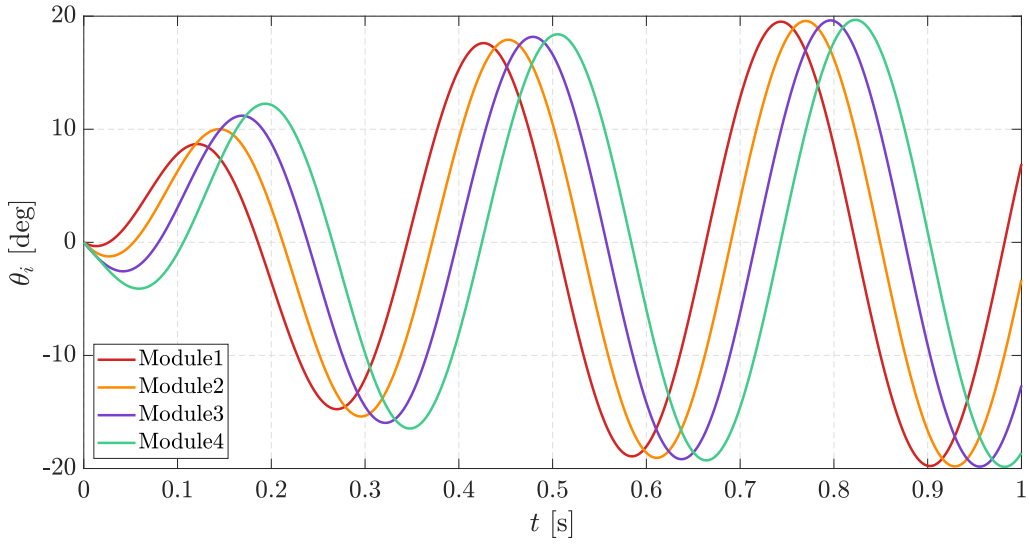
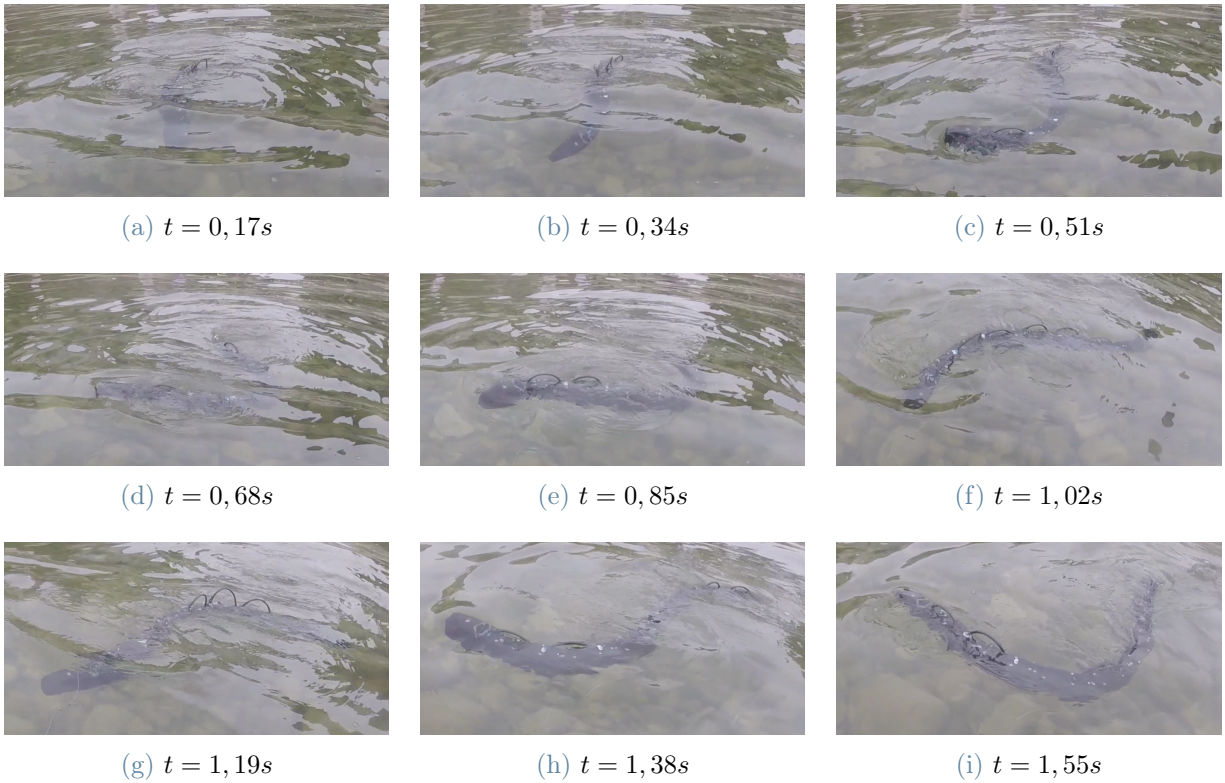


Figure 7.1: Reference signal passed to first 4 modules to perform lateral undulation task



(a) $t = 0,17s$ (b) $t = 0,34s$ (c) $t = 0,51s$

(d) $t = 0,68s$ (e) $t = 0,85s$ (f) $t = 1,02s$

(g) $t = 1,19s$ (h) $t = 1,38s$ (i) $t = 1,55s$

Figure 7.2: Test with lateral undulation, each time of the frame is specified under the image, $f = 0,5\text{Hz}$, $A = 20°$, $\varphi = \frac{\pi}{6}\text{rad}$

7.1.2. Eel-like motion

While performing eel-like motion the whole robot tends to sink more than with lateral undulation. This is probably due to the fact that eel-like have overall amplitudes that are smaller with respect to lateral undulation, according to equation 6.5, so in order to achieve higher velocities parameters such as frequency and amplitude must be increased. However, tests for such values were not performed, due to our decision to focus on lateral undulation behavior. Reference signals for the test are shown in Figure 7.3 while frames of the experiment's video are shown in Figure 7.4.

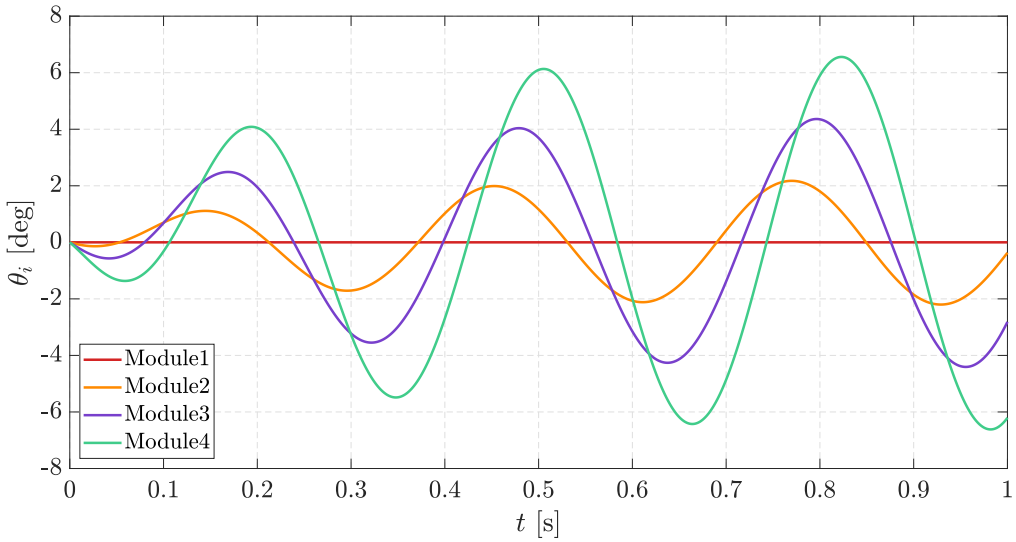
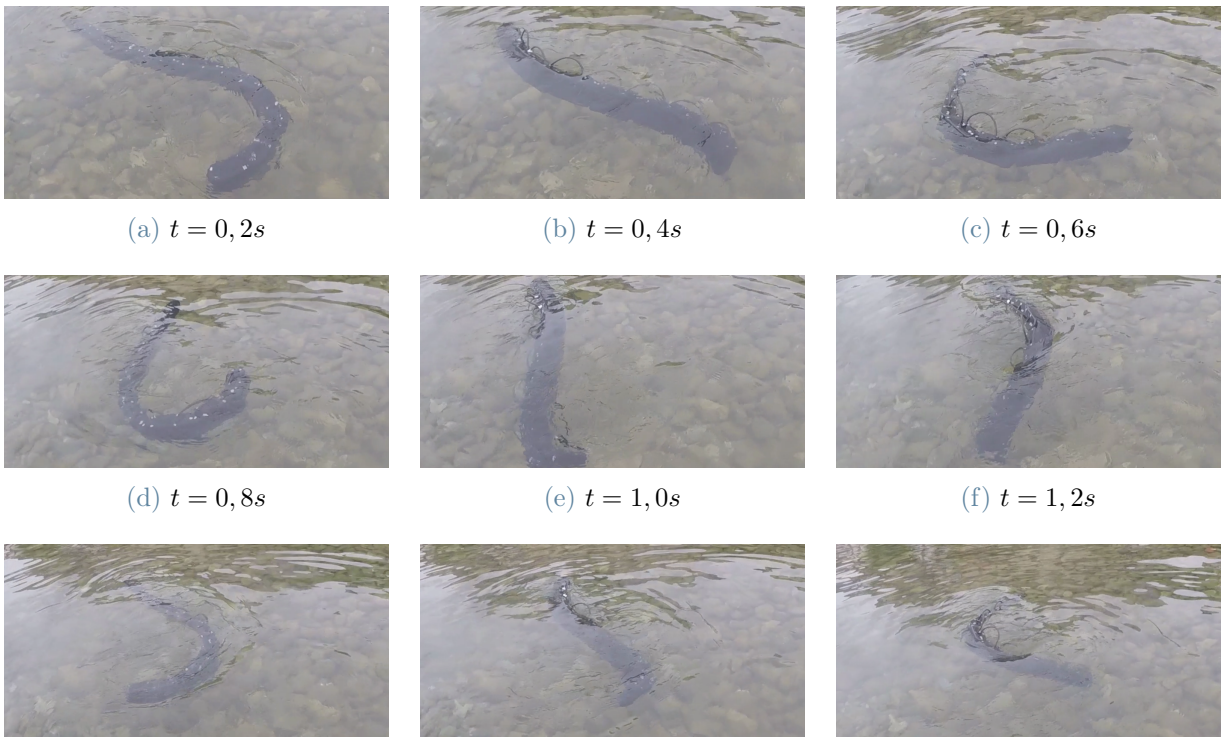


Figure 7.3: Reference signal passed to first 4 modules to perform eel-like motion



(a) $t = 0,2s$ (b) $t = 0,4s$ (c) $t = 0,6s$

(d) $t = 0,8s$ (e) $t = 1,0s$ (f) $t = 1,2s$

(g) $t = 1,4s$ (h) $t = 1,6s$ (i) $t = 1,8s$

Figure 7.4: Experimental test with eel-like motion, each time of the frame is specified under the image, $f = 0,5\text{Hz}$, $A = 20°$, $\varphi = \frac{\pi}{6}\text{rad}$

7.1.3. Steering

The steering operation was performed with the forward swimming parameters listed in Table 7.1. The main feature that has been discovered is that the constant-hold strategy is the best one in terms of pure steering and energy saving, while the constant offset is better in terms of speed. The experiment for the constant offset method is shown in Figure 7.6 while its reference signal in 7.5. The experiment for the constant hold method is shown in Figure 7.8 while its reference signal in 7.7.

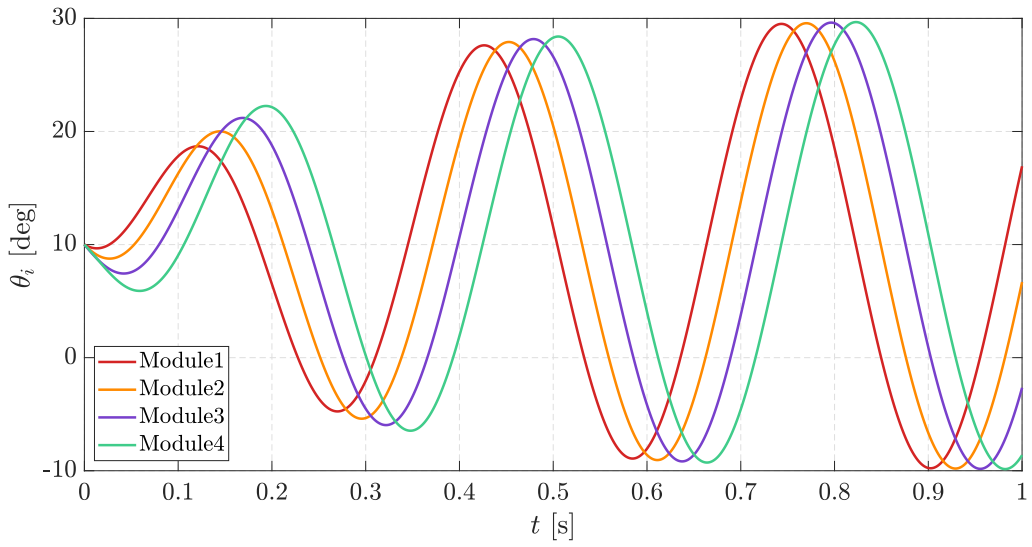
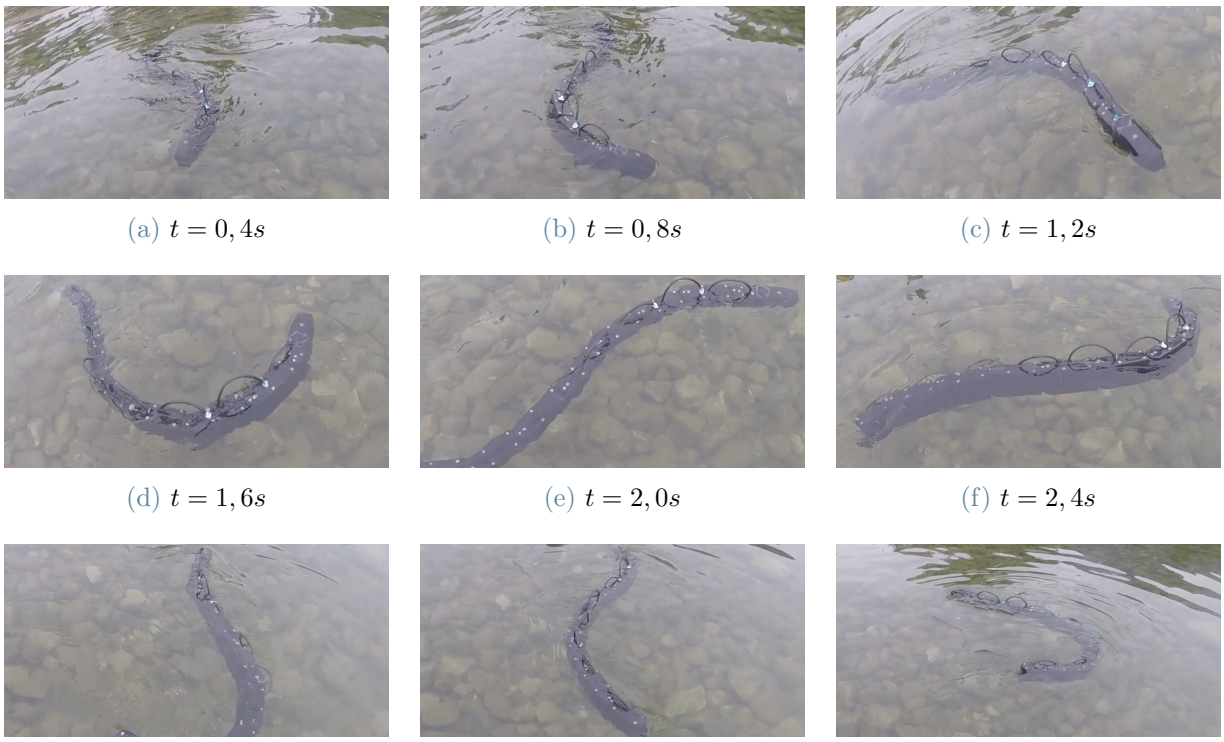


Figure 7.5: Reference signal passed to first 4 modules to perform steering in constant offset mode



(a) $t = 0,4s$ (b) $t = 0,8s$ (c) $t = 1,2s$

(d) $t = 1,6s$ (e) $t = 2,0s$ (f) $t = 2,4s$

(g) $t = 2,8s$ (h) $t = 3,2s$ (i) $t = 3,6s$

Figure 7.6: Experimental test for steering with constant offset method, each time of the frame is specified under the image, $f = 0,5\text{Hz}$, $A = 20°$, $\varphi = \frac{\pi}{6}\text{rad}$

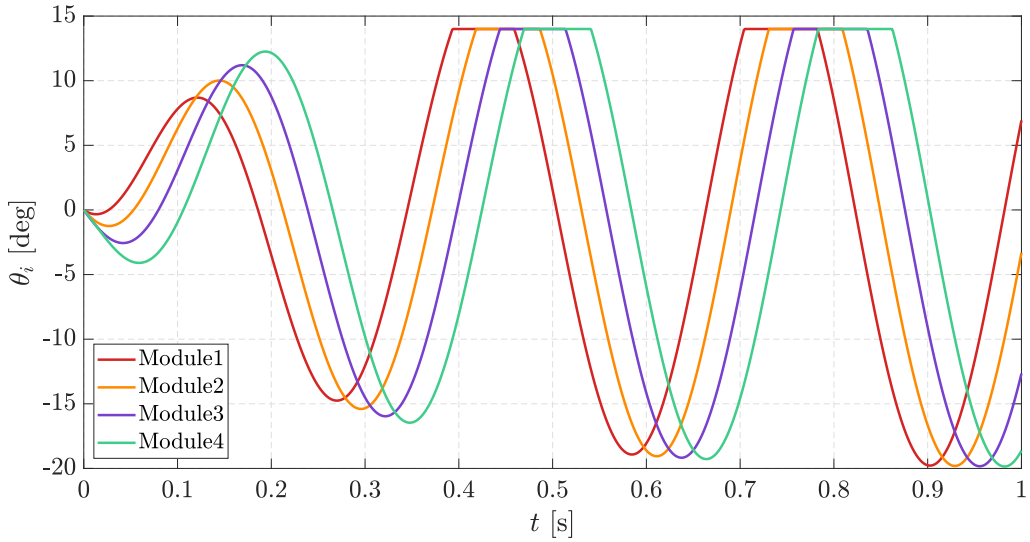
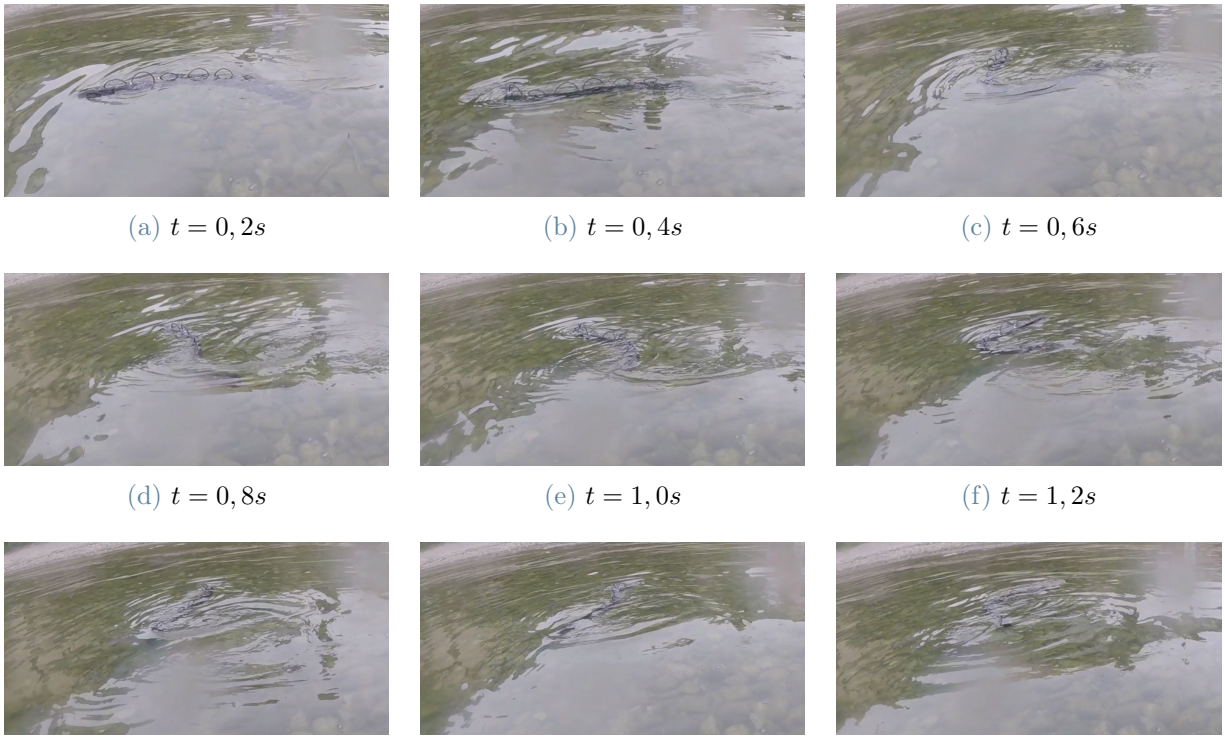


Figure 7.7: Reference signal passed to first 4 modules to perform steering in constant hold mode



(a) $t = 0,2s$ (b) $t = 0,4s$ (c) $t = 0,6s$

(d) $t = 0,8s$ (e) $t = 1,0s$ (f) $t = 1,2s$

(g) $t = 1,4s$ (h) $t = 1,6s$ (i) $t = 1,8s$

Figure 7.8: Experimental test for steering with constant hold method, each time of the frame is specified under the image, $f = 0,5\text{Hz}$, $A = 20°$, $\varphi = \frac{\pi}{6}\text{rad}$

7.1.4. Obstacle avoidance

During experiments, some parameters had to be changed, minimum distance had to be shortened in order to take into account wave's propagation in aquatic environment. According to the law

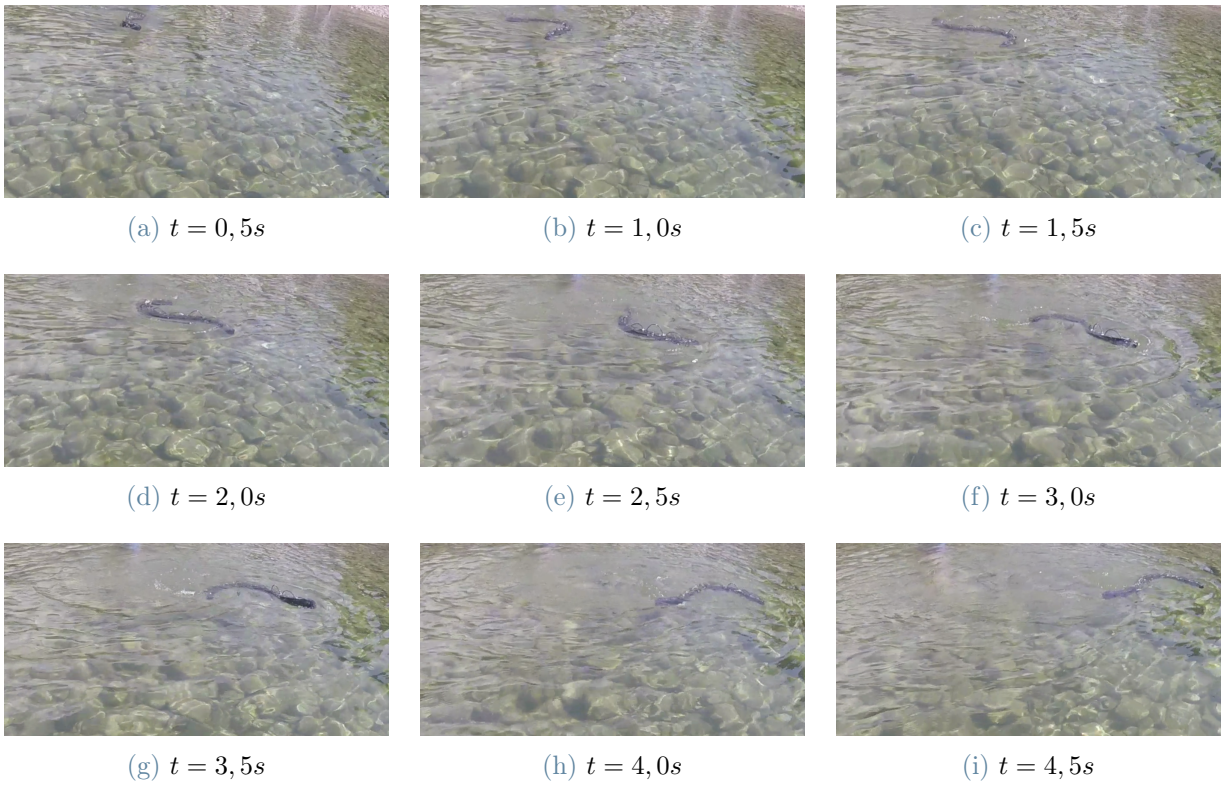
$$v = \sqrt{\frac{\kappa}{\rho}}, \tag{7.1}$$

where κ is the compressibility module of the medium and ρ its density, the ratio between wave's propagating speed in water and air has been calculated as

$$\bar{v} = \sqrt{\frac{\kappa_{water}\rho_{air}}{\kappa_{air}\rho_{water}}} = 4,33 \tag{7.2}$$

By dividing the value of the minimum distance with the factor $\bar{v}$ we obtained the desired value. Also the value of T was reduced, which was discussed in Equations 6.9 and 6.10, in order to have faster response from ultrasonic sensor's data. With those parameters, the robot was capable of completely avoiding any number of obstacles on any trajectory it followed. Figure 7.9 shows the robot while it is performing lateral undulation on a straight

line and avoids the obstacle represented by the camera's operator when it is 150cm distant from him.



(a) $t = 0,5s$ (b) $t = 1,0s$ (c) $t = 1,5s$

(d) $t = 2,0s$ (e) $t = 2,5s$ (f) $t = 3,0s$

(g) $t = 3,5s$ (h) $t = 4,0s$ (i) $t = 4,5s$

Figure 7.9: Experimental test for obstacle avoidance, each time of the frame is specified under the image, $f = 0,5\text{Hz}$, $A = 20°$, $\varphi = \frac{\pi}{6}\text{rad}$

7.2. Final tests and simulation's validation

In this section, a discussion regarding the simulation and its use for detecting optimal parameters is performed. From the first experimental test with lateral undulation a value of the speed of $21\frac{cm}{s}$ was measured, while for the same parameters, a value of the speed of $18,79\frac{cm}{s}$ was found with the simulation approach. Knowing that the use of a linear approximation of the motor's characteristic curve should lead to simulation data for the speed that are underestimated, the above-measured data can be considered to be similar. At this point, it is interesting to test parameters in correspondence of the red dot in Figure 7.10, where the value of the maximum speed should be found. Therefore, a test considering the set of parameters reported in Table 7.2 was performed, obtaining a speed of $34,47\frac{cm}{s}$, which is again different but still coherent with the assumption made. Frames of the final test are shown in Figrue 7.12.

	Value	**Unit**
A	30	deg
f	$0,9$	Hz
φ	$0,9$	rad

Table 7.2: List of parameters for finding maximum speed with lateral undulation

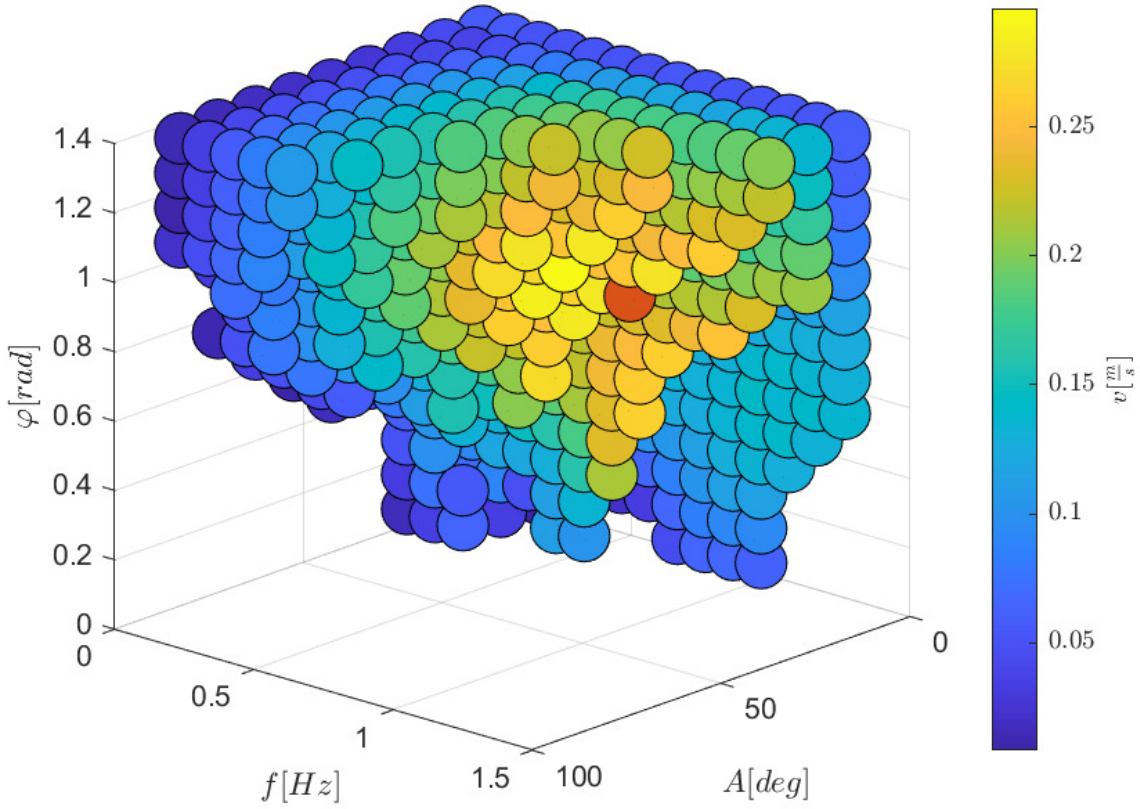


Figure 7.10: Simulation's result

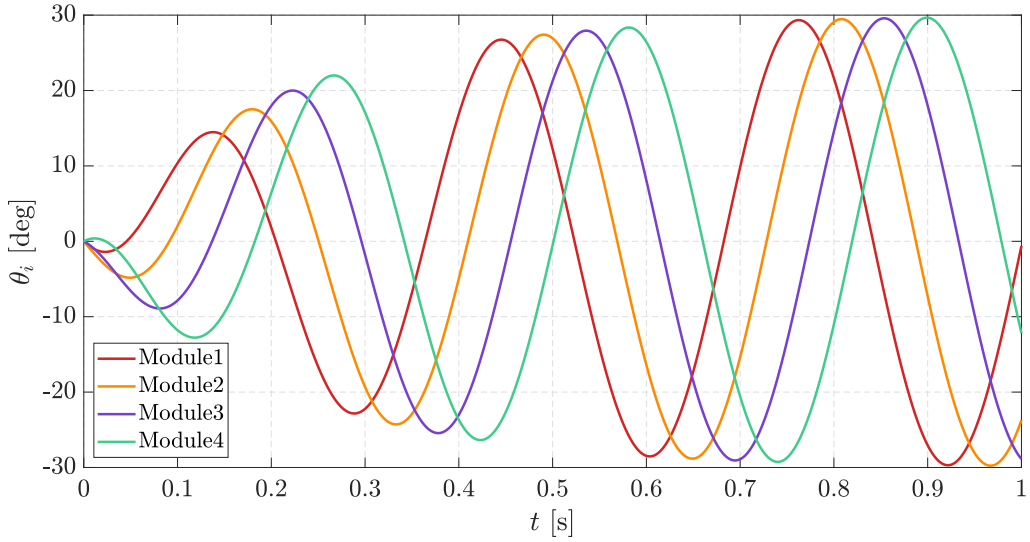
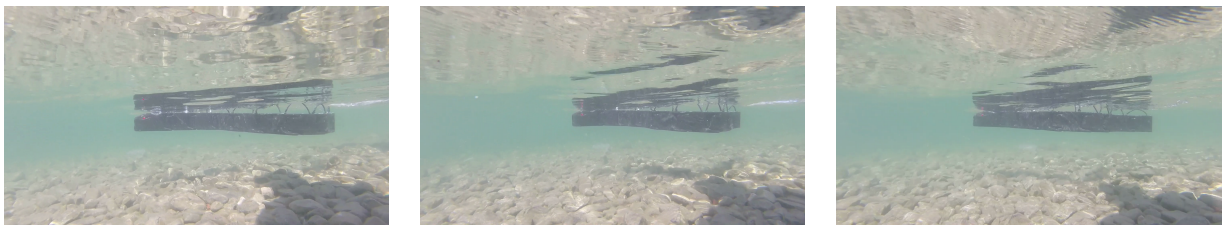


Figure 7.11: Reference signal passed to first 4 modules to perform maximum speed lateral undulation

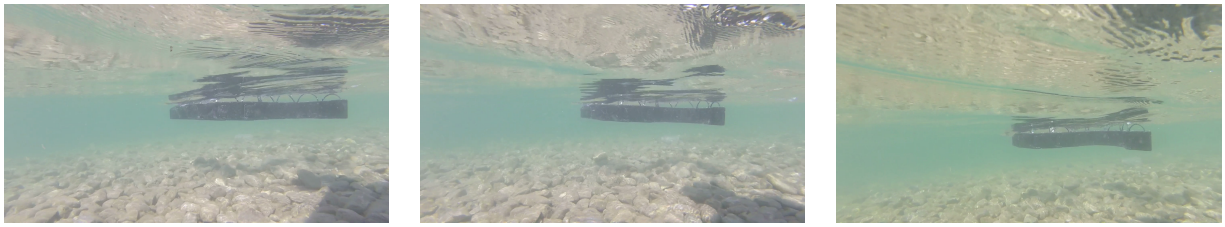
(a) $t = 0,4s$ (b) $t = 0,8s$ (c) $t = 1,2s$

(d) $t = 1,6s$ (e) $t = 2,0s$ (f) $t = 2,4s$

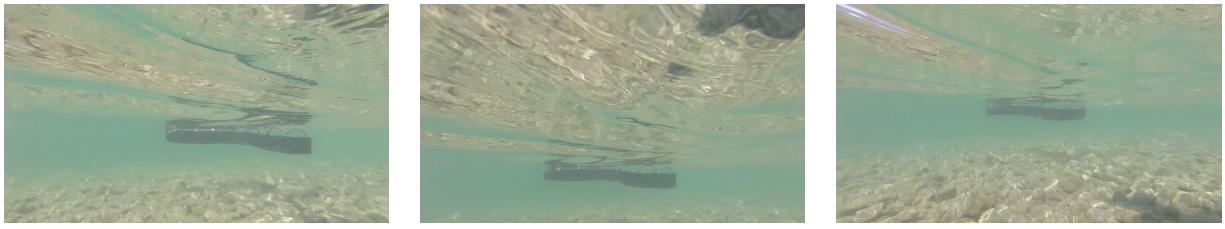
(g) $t = 2,8s$ (h) $t = 3,2s$ (i) $t = 3,6s$



(j) $t = 0,4s$ (k) $t = 0,8s$ (l) $t = 1,2s$



(m) $t = 1,6s$ (n) $t = 2,0s$ (o) $t = 2,4s$



(p) $t = 2,8s$ (q) $t = 3,2s$ (r) $t = 3,6s$

Figure 7.12: Experimental test for maximum speed obtained from the two different prospective, each time of the frame is specified under the image, $f = 0,9\text{Hz}$, $A = 30°$, $\varphi = 0,9\text{rad}$

8 | Conclusions and future developments

The older version of the robot was tested and limitations emerged. The old version was not capable of obtaining higher values of the speed with respect to the one that it was already capable of. The excessive buoyancy of the robot created by the cover was an obstacle to its movement and made it difficult to test the real behavior of the sinusoidal movement. The new version of the robot has lower inertia, more powerful motors, more precise low-level control, practical switch and charging operations, external encoders and the possibility to save data inside a micro SD. The robot is capable of reaching speeds that are much higher with respect to the previous version and it is also capable of avoiding obstacles with the integrated feedback control realized using the ultrasonic sensor. The overall robot resulted to be more practical from the user's perspective and was completely waterproof, in fact, no damage or practical issues emerged from the presence of water during the experiments. The simulation effectively handles the interaction between the fluid and the robot's structure. The speed and torque values derived from this simulation have been tested and demonstrate coherence when evaluated within the context of the underlying hypotheses. Moreover, the obtained data are collected in a shorter amount of time with respect to other strategies.

For the future version, a more robust feedback control system can be built. Through the use of IMU's feedback, already present inside this version of the robot, head orientation control can be included. Moreover, with the addition of a compass or GPS, trajectory planning tasks can be developed. CPG algorithm can be implemented to improve the generation of the sinusoidal motion law, due to their particular predisposition in managing sinusoidal signals. Detailed studies have to be performed regarding the difference between simulation and experimental results. The emerged trend is obtained for a linearly approximated servo motor characteristic curve: for this reason, the real robot tested in the same conditions could achieve higher speeds, and, therefore, a more sophisticated approximation of the characteristic curve of the servo-motor should be introduced. The new

approximation could discover different high-performance zones. Moreover, increasing the computational power available could produce more precise results and simulate a higher liquid volume to better represent the real situation.

Bibliography

[1] Dse dry suit experience, 2023. URL https://www.drysuitexperience.com/.

[2] Important classes - monobehaviour, 2023. URL https://docs.unity3d.com/Manual/class-MonoBehaviour.html.

[3] Physics, 2023. URL https://docs.unity3d.com/Manual/PhysicsSection.html.

[4] Plug-ins, 2023. URL https://docs.unity3d.com/Manual/Plugins.html.

[5] Articulation body component reference, 2023. URL https://docs.unity3d.com/Manual/class-ArticulationBody.html.

[6] Zibra liquids, 2023. URL https://assetstore.unity.com/packages/tools/physics/zibra-liquids-200718.

[7] 2023. URL https://www.inaturalist.org/taxa/35164-Hydrophis-platurus/browse_photos.

[8] F. Alvarez and A. Celis. On the occurrence of conchoderma virgatum and dosima fascicularis (cirripedia, thoracica) on the sea snake, pelamis platurus (reptilia, serpentes) in jalisco, mexico. *Crustaceana*, 77(6):761–764, 2004.

[9] Z. Y. Bayraktaroglu. Snake-like locomotion: Experimentations with a biologically inspired wheel-less snake robot. *Mechanism and Machine Theory*, 44(3):591–602, 2009.

[10] J. Bonet and R. D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 2 edition, 2008. doi: 10.1017/CBO9780511755446.

[11] I.-L. G. Borlaug, K. Y. Pettersen, and J. T. Gravdahl. Comparison of two second-order sliding mode control algorithms for an articulated intervention auv: Theory and experimental results. *Ocean Engineering*, 222:108480, 2021.

[12] F. Brischoux and R. Shine. Morphological adaptations to marine life in snakes. *Journal of morphology*, 272(5):566–572, 2011.

[13] J. A. Campbell, W. W. Lamar, E. D. Brodie, et al. *The venomous reptiles of the western hemisphere*, volume 1. Comstock Pub. Associates Ithaca [NY], 2004.

[14] K. E. Carpenter, V. H. Niem, et al. *FAO species identification guide for fishery purposes. The living marine resources of the Western Central Pacific. Volume 6. Bony fishes part 4 (Labridae to Latimeriidae), estuarine crocodiles, sea turtles, sea snakes and marine mammals.* FAO Library, 2001.

[15] S. Chigisaki, M. Mori, H. Yamada, and S. Hirose. Design and control of amphibious snake-like robot acm-r5. *Nippon Kikai Gakkai Robotikusu, Mekatoronikusu Koenkai Koen Ronbunshu (CD-ROM)*, 43, 2005.

[16] F. Cooke and J. Bruce. *The encyclopedia of animals: a complete visual guide*. Univ of California Press, 2004.

[17] A. Crespi and A. J. Ijspeert. Amphibot ii: An amphibious snake robot that crawls and swims using a central pattern generator. In *Proceedings of the 9th international conference on climbing and walking robots (CLAWAR 2006)*, number CONF, pages 19–27, 2006.

[18] A. Crespi, A. Badertscher, A. Guignard, and A. J. Ijspeert. Amphibot i: an amphibious snake-like robot. *Robotics and Autonomous Systems*, 50(4):163–175, 2005.

[19] A. de Vaucorbeil, V. P. Nguyen, S. Sinaie, and J. Y. Wu. Chapter two - material point method after 25 years: Theory, implementation, and applications. volume 53 of *Advances in Applied Mechanics*, pages 185–398. Elsevier, 2020. doi: https://doi.org/10.1016/bs.aams.2019.11.001. URL `https://www.sciencedirect.com/science/article/pii/S0065215619300146`.

[20] E. Gautreau, X. Bonnet, J. Sandoval, G. Fosseries, A. Herrel, M. Arsicault, S. Zeghloul, and M. A. Laribi. A biomimetic method to replicate the natural fluid movements of swimming snakes to design aquatic robots. *Biomimetics*, 7(4):223, 2022.

[21] E. Gautreau, X. Bonnet, T. Fox, G. Fosseries, V. Valle, A. Herrel, and M. A. Laribi. Complementary methods to acquire the kinematics of swimming snakes: a basis to design bio-inspired robots. *Journal of Bionic Engineering*, 20(2):668–682, 2023.

[22] A. Geographic. Sea snake robot to help study marine life, 2014. URL `https://www.australiangeographic.com.au/news/2014/02/sea-snake-robot-to-help-study-marine-life/`.

[23] G. B. Gillis. Anguilliform locomotion in an elongate salamander (siren intermedia):

effects of speed on axial undulatory movements. *Journal of Experimental Biology*, 200(4):767–784, 1997.

[24] J. B. Graham, W. R. Lowell, I. Rubinoff, and J. Motta. Surface and subsurface swimming of the sea snake pelamis platurus. *Journal of Experimental Biology*, 127 (1):27–44, 1987.

[25] S. Hirose. Biologically inspired robots. *Snake-Like Locomotors and Manipulators*, 1993.

[26] J. K. Hopkins, B. W. Spranklin, and S. K. Gupta. A survey of snake-inspired robot designs. *Bioinspiration & biomimetics*, 4(2):021001, 2009.

[27] Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, and C. Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. Graph.*, 37(4), jul 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201293. URL `https://doi.org/10.1145/3197517.3201293`.

[28] T. J. Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[29] M. Hultmark, M. Leftwich, and A. J. Smits. Flowfield measurements in the wake of a robotic lamprey. *Experiments in fluids*, 43:683–690, 2007.

[30] P. Hutchings, M. Kingsford, and O. Hoegh-Guldberg. *The Great Barrier Reef: biology, environment and management*. Csiro publishing, 2019.

[31] D. Iseli. How do snakes swim? amazing!, 2021. URL `https://www.animalfoodplanet.com/how-do-snakes-swim/`.

[32] M. S. M. Jasni, R. E. Samin, and B. S. K. Ibrahim. Biological inspired inspection underwater robot (snakey). *Procedia Engineering*, 41:1058–1064, 2012.

[33] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, SIGGRAPH '16, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342896. doi: 10.1145/2897826.2927348. URL `https://doi.org/10.1145/2897826.2927348`.

[34] E. Kelasidi, P. Liljebäck, K. Y. Pettersen, and J. T. Gravdahl. Experimental investigation of efficient locomotion of underwater snake robots for lateral undulation and eel-like motion patterns. *Robotics and Biomimetics*, 2:1–27, 2015.

[35] E. Kelasidi, K. Y. Pettersen, A. M. Kohl, and J. T. Gravdahl. An experimental

investigation of path following for an underwater snake robot with a caudal fin. *IFAC-PapersOnLine*, 50(1):11182–11190, 2017.

[36] M. S. U. Khalid, J. Wang, I. Akhtar, H. Dong, M. Liu, and A. Hemmati. Why do anguilliform swimmers perform undulation with wavelengths shorter than their bodylengths? *Physics of Fluids*, 33(3), 2021.

[37] H. Kimura and S. Hirose. Development of genbu: Active wheel passive joint articulated mobile robot. In *IEEE/RSJ international conference on intelligent robots and systems*, volume 1, pages 823–828. IEEE, 2002.

[38] A. M. Kohl, E. Kelasidi, A. Mohammadi, M. Maggiore, and K. Y. Pettersen. Planar maneuvering control of underwater snake robots using virtual holonomic constraints. *Bioinspiration & biomimetics*, 11(6):065005, 2016.

[39] A. Kuwada, S. Wakimoto, K. Suzumori, and Y. Adomi. Automatic pipe negotiation control for snake-like robot. In *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 558–563. IEEE, 2008.

[40] J. C. Liao. Swimming in needlefish (belonidae): anguilliform locomotion with fins. *Journal of Experimental Biology*, 205(18):2875–2884, 2002.

[41] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl. *Snake robots: modelling, mechatronics, and control*. Springer, 2013.

[42] P. Liljebäck, Ø. Stavdahl, K. Y. Pettersen, and J. T. Gravdahl. Mamba-a waterproof snake robot with tactile sensing. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 294–301. IEEE, 2014.

[43] P. Liljebäck, K. Pettersen, Stavdahl, and J. Gravdahl. A review on modelling, implementation, and control of snake robots. *Robotics and Autonomous Systems*, 60(1): 29–40, 2012. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2011.08.010. URL `https://www.sciencedirect.com/science/article/pii/S0921889011001618`.

[44] J. M. Mehrtens. Living snakes of the world in color. *(No Title)*, 1987.

[45] K. Melsaac and J. Ostrowski. A geometric approach to anguilliform locomotion: modelling of an underwater eel robot. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 4, pages 2843–2848 vol.4, 1999. doi: 10.1109/ROBOT.1999.774028.

[46] A. Ming, T. Ichikawa, W. Zhao, and M. Shimojo. Development of a sea snake-

like underwater robot. In *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, pages 761–766. IEEE, 2014.

[47] S. A. Minton. Lethal toxicity of venoms of snakes from the coral sea. *Toxicon*, 21 (6):901–902, 1983.

[48] M. Mori and S. Hirose. Development of active cord mechanism acm-r3 with agile 3d mobility. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 3, pages 1552–1557. IEEE, 2001.

[49] K. P. H. H. Mudiyanselage. Design, realization and control of an aquatic snake robot. Master's thesis, Politecnico di Milano, 2022.

[50] U. K. MULLER, J. Smit, E. J. Stamhuis, and J. J. Videler. How the body contributes to the wake in undulatory fish swimming: flow fields of a swimming eel (anguilla anguilla). *Journal of Experimental Biology*, 204(16):2751–2762, 2001.

[51] Oceana. Watch: Mesmerizing ribbon eels swim gracefully | oceana, 2020. URL `https://www.youtube.com/watch?v=ofLpeY3xdNA`.

[52] H. Ohno and S. Hirose. Study on slime robot (proposal of slime robot and design of slim slime robot). In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, volume 3, pages 2218–2223. IEEE, 2000.

[53] K. Y. Pettersen. Snake robots. *Annual Reviews in Control*, 44:19–44, 2017. ISSN 1367-5788. doi: https://doi.org/10.1016/j.arcontrol.2017.09.006. URL `https://www.sciencedirect.com/science/article/pii/S1367578817301050`.

[54] M. Sfakiotakis, D. M. Lane, and J. B. C. Davies. Review of fish swimming modes for aquatic locomotion. *IEEE Journal of oceanic engineering*, 24(2):237–252, 1999.

[55] E. Sherratt, A. R. Rasmussen, and K. L. Sanders. Trophic specialization drives morphological evolution in sea snakes. *Royal Society open science*, 5(3):172141, 2018.

[56] C. Stefanini, S. Orofino, L. Manfredi, S. Mintchev, S. Marrazza, T. Assaf, L. Capantini, E. Sinibaldi, S. Grillner, P. Wallen, et al. A compliant bioinspired swimming robot with neuro-inspired control and autonomous behavior. In *2012 IEEE International Conference on Robotics and Automation*, pages 5094–5098. IEEE, 2012.

[57] M. Steffen, R. M. Kirby, and M. Berzins. Analysis and reduction of quadrature errors in the material point method (mpm). *International Journal for Numerical Methods*

in Engineering, 76(6):922–948, 2008. doi: https://doi.org/10.1002/nme.2360. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2360`.

[58] J. Stidworthy, D. McDougal, et al. *Snakes of the world by John Stidworthy; illustrated by Dougal MacDougal*. Grosset & Dunlap, 1974.

[59] J. Sverdrup-Thygeson, E. Kelasidi, K. Y. Pettersen, and J. T. Gravdahl. The underwater swimming manipulator-a bio-inspired auv. In *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 387–395. IEEE, 2016.

[60] R. Thandiackal, K. Melo, L. Paez, J. Herault, T. Kano, K. Akiyama, F. Boyer, D. Ryczko, A. Ishiguro, and A. J. Ijspeert. Emergence of robust self-organized undulatory swimming based on local hydrodynamic force sensing. *Science Robotics*, 6 (57):eabf6354, 2021. doi: 10.1126/scirobotics.abf6354. URL `https://www.science.org/doi/abs/10.1126/scirobotics.abf6354`.

[61] E. D. Tytell. The hydrodynamics of eel swimming ii. effect of swimming speed. *Journal of experimental biology*, 207(19):3265–3279, 2004.

[62] E. D. Tytell and G. V. Lauder. The hydrodynamics of eel swimming: I. wake structure. *Journal of Experimental Biology*, 207(11):1825–1841, 2004.

[63] K. Wang, W. Gao, and S. Ma. Snake-like robot with fusion gait for high environmental adaptability: Design, modeling, and experiment. *Applied Sciences*, 7(11):1133, 2017.

[64] X. Wang, Z. Yuan, Q. Guo, W. Wang, H. Liu, A. Liu, Z. Ge, H. Yu, and W. Yang. An underwater bionic snake soft robot with tunable deformation and motion based on composite materials. *Advanced Materials Technologies*, page 2202012, 2023.

[65] R. Worst and R. Linnemann. Construction and operation of a snake-like robot. In *Proceedings IEEE international joint symposia on intelligence and systems*, pages 164–169. IEEE, 1996.

[66] C. Wright, A. Johnson, A. Peck, Z. McCord, A. Naaktgeboren, P. Gianfortoni, M. Gonzalez-Rivero, R. Hatton, and H. Choset. Design of a modular snake robot. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2609–2614. IEEE, 2007.

[67] C. Wright, A. Buchan, B. Brown, J. Geist, M. Schwerin, D. Rollinson, M. Tesch, and H. Choset. Design and architecture of the unified modular snake robot. In *2012 IEEE international conference on robotics and automation*, pages 4347–4354. IEEE, 2012.

[68] H. Yamada and S. Hirose. Development of practical 3-dimensional active cord mechanism acm-r4. *Journal of Robotics and Mechatronics*, 18(3):305–311, 2006.

[69] W. Zhu, X. Guo, and Y. Fang. Design of a modular snake robot and control with internet of things. In *2017 Chinese Automation Congress (CAC)*, pages 850–854. IEEE, 2017.

[70] Z. Zuo, Z. Wang, B. Li, and S. Ma. Serpentine locomotion of a snake-like robot in water environment. In *2008 IEEE International Conference on Robotics and Biomimetics*, pages 25–30. IEEE, 2009.

List of Figures

List of Tables

Acknowledgements

We would like to thank Prof. Simone Cinquemani for giving us the opportunity to work on this project. Special thanks go to Giovanni Bianchi, who oversaw our work from the origin: without its precious support this adventure would not have been possible.