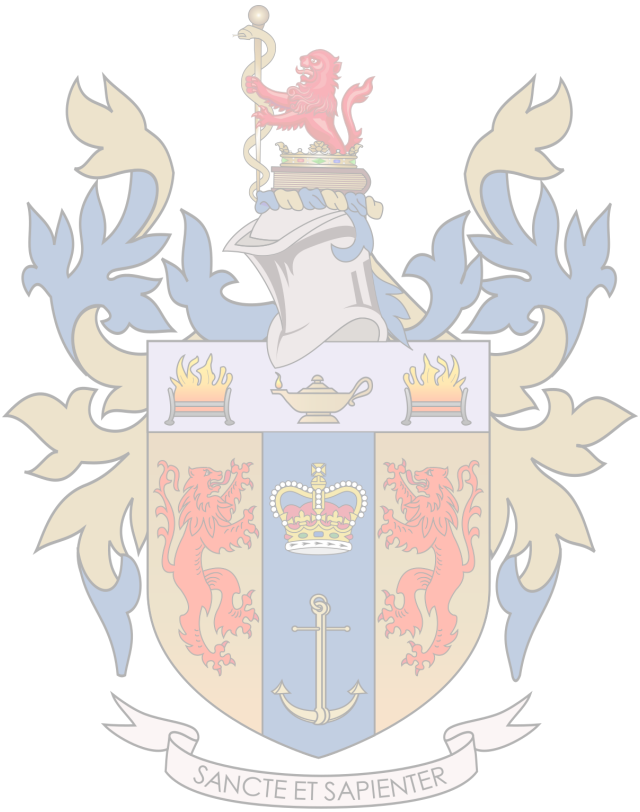
Autonomous Underwater Glider Drone for Environmental Monitoring of Lakes

Sophie Caplan, William Droin, Rohith Raghunathan Nair,
Doran Moison, Dhruv Pandit, Arushi Khokhar

King's College London, Department of Natural, Mathematical & Engineering Sciences
MSc in Robotics
16th April 2024



Contents

1 Product Brief

1.1 About

The Autonomous Underwater Glider represents a significant advancement in the field of lake health monitoring, addressing the inherent limitations of traditional surveying methods. Engineered to navigate lake depths with exceptional efficiency, it offers extensive temporal and spatial coverage while optimising resources. At its core, the glider operates autonomously, eliminating the need for costly human intervention and ensuring continuous data collection. Equipped with advanced sensors, it captures a wide range of data, including water quality metrics, ecosystem dynamics, and critical environmental changes necessary for effective mitigation strategies. In contrast to static sensors with limited adaptability or crew-dependent monitoring that consumes resources, the glider provides a cost-effective alternative with long-endurance capabilities. This not only improves the frequency and accuracy of data collection but also optimises resource allocation for sustainable lake monitoring practices.

1.2 Key Elements

Figure 1: Detailed CAD model of the autonomous underwater glider. The glider is depicted in a static position, showcasing its main components and key features.

1.3 Technical Specifications

Table 1 details the technical specifications of the glider.

Table 1: Technical Specifications of the Autonomous Underwater Glider Components

Component	Value
General	
Dry Weight	31.934 Kg
Total Length	1970mm
Hull	
Length	1700mm
Outer diameter	150mm
Inner diameter	140mm
Material	Acrylic
Control System and Sensors	
Linear Actuators x2	12v, 3.5A, 300mm (Length), 25% (Duty Cycle)
Motor Drivers x2	BTS7960B (5v, 43A)
Temperature Sensor	TSYS01 (3.3v), Range: -40 to +125 C, Resolution: 0.1 C
Sonar	Ping Sonar (3.3v), Range: 0.3 - 100m, Resolution: 0.001m
Microcontroller	Arduino Uno R4, Core: ESP32-S3, Connectivity: USB-C, WiFi, Bluetooth
Buoyancy Engine	
Syringe x2	Volume: 200ml
Syringe Holder, PLA	130mm x 57mm (Outer Diameter X Depth)
Syringe Holder Base, PLA	130mm x 25mm (Outer Diameter X Depth)
Secure Pin, Wood	130mm (Length)
Tube	36cm
Battery/Pitch Control Module	
NiMH Battery (D)	1.2v, 10000mAH
Pentagonal Prism, PLA	130mm x 40mm (Depth X Side)
Pentagon End Supports x2	20mm x 40mm (Depth X Side)
Metal Springs	10 units
Rectangular Battery base x5	150mm x 39.5mm x 3mm (Length X Width X Thickness)
Battery Fastener	30 units
Wings	
Aluminium Wings x2	600mm x 200mm (Length X Width)
Wing Mounts (Upper and Lower)	270mm x 200mm (Length X Width)
Bolts	6
Nuts	6
Other	
End caps x2, Rubber	165mm x 155mm (Outer diameter X Inner diameter)
Nose	250mm
Reinforcing Mesh, x9	130mm x 130.5mm (Outer diameter X Inner diameter)
Mesh Plugs	19.5mm x 4mm (Outer diameter X Inner diameter)

1.4 Mission Statement

The project's mission is to revolutionise lake health monitoring through the development and deployment of an autonomous underwater glider. It aims to address significant challenges faced by traditional surveying methods by providing a comprehensive and cost-effective solution that navigates lake depths with exceptional efficiency. The glider's autonomy ensures continuous data collection, empowering researchers, environmental agencies, and policymakers with actionable insights crucial

Figure 2: Alpha version of the autonomous underwater glider by the Mercers Lake in Redhill, UK

for preserving and restoring invaluable lake ecosystems worldwide. The project's innovative approach strives to bridge gaps in current data collection methods, ushering in a new era of sustainable and informed decision-making for the benefit of the planet's vital aquatic ecosystems.

1.5 Conclusion

The Autonomous Underwater Glider represents a groundbreaking leap in lake health monitoring, overcoming the limitations of conventional survey methods. Engineered for efficient navigation of lake depths, it provides extensive spatial and temporal coverage while optimising resource utilisation. Its autonomous operation eliminates costly human intervention, ensuring uninterrupted data collection. The glider can be customised to incorporate advanced sensors, enabling it to capture a diverse array of data critical for effective mitigation strategies and environmental preservation. This glider offers a cost-effective alternative to static sensors or crew-dependent monitoring, improving data frequency and accuracy while optimising resource allocation for sustainable lake monitoring practices. Figure 2 shows the prototype of the autonomous underwater glider following its testing in the waters of Mercers Lake in Redhill, UK. The video showcasing the glider's successful ascent and descent in the controlled environment of the ocean towing tank at University College London **can be viewed here**. This demonstration effectively shows the glider's functionality.

2 Appendices

2.1 Follow Up on Project Plan

Having created a project plan in November allowed for a clear sense of direction throughout the project. Many important decisions about the glider drone's functionalities and material choices had been made or reviewed during the pre-study, streamlining work down the line. Potential setbacks were identified, and the technical and administrative challenges and ethical considerations were explained. However, as the project progressed, deviations emerged, necessitating agile adjustments to initial plans.

One of the first re-considerations was regarding the drone's hull. Initially planning to use a standard PVC tube, no pipes with a large enough diameter for the drone were found. As an alternative, an acrylic tube was chosen, still suited to the product due to its durability, impact resistance, and relative affordability. Despite PVC offering superior qualities for the glider drone's hull, being more durable and heavier than acrylic, supply limitations required the use of the latter.

While an Arduino Nano ESP32 was preferred in the initial stages of the project, connectivity limitations with the sensors meant that an Arduino Uno had to be used. This had been considered during the pre-study, however, instead of using the ESP32 for controlling the glider's navigation and the Uno for reading analogue data from the sensors, it was decided that the Arduino Uno R4 Wi-Fi would handle all functionalities. As per the project proposal, the Arduino programming language was used, which is based on C++. The Arduino IDE was used for writing and uploading code to the Arduino boards and monitoring during tests.

The original plan included a wide array of sensors for the drone, serving both glider control and data-gathering purposes. However, administrative limitations concerning approved suppliers necessitated a reevaluation of sensor choices. Ultimately, a sonar and an IMU were obtained for the control systems, along with a temperature sensor for environmental monitoring. With test deadlines approaching, time constraints required a decision on which sensors to implement. Opting for a minimum viable product (MVP), only the core features of the drone were included. Consequently, the temperature sensor was deemed non-essential, and only the sonar and IMU were used.

Initial research and technical requirements had indicated a preference for a rechargeable 12V lithium-ion battery for the glider's power supply. While the lithium-ion (Li-ion) battery was deemed the best overall choice, administrative limitations regarding suppliers and safety regulations prevented its use. Instead, nickel-metal hydride (NiMH) batteries were selected, despite Li-ion batteries offering higher energy density, longer cycle life, faster charging times, and lower self-discharge rates [1], [2], as well as a lower overall environmental impact [3]. The decision to switch the motor drivers from the L298N to the BTS7960B was taken because of the inability of the former to meet the power demands of the linear actuators. With each actuator drawing approximately 3.5A, it was determined that the BTS7960B motor driver, rated for a minimum peak current capability of 33A, was better suited for the task.

In the aim of creating a minimum viable product, several functionalities deemed unnecessary for the first version of the product were excluded. While the buoyancy control loop was implemented, path planning and course correction were omitted since navigation was not required for the final tests. With the removal of environmental sensors, there was no need to store collected data in a database, however also resulting in the inability to perform any data preprocessing. Although these functionalities were excluded from the final product, they can be easily added by uploading the respective code and mounting the sensors onto the drone. Modularity was a key aspect of the project and is present in both the drone's structure and its code; the inner mesh allows for easy interchangeability of modules and modular programming was used while developing the code. Minimal or no modifications to the glider's structure would be required for implementation of additional functionalities.

Numerous tests were performed to ensure a fully operational product by the project's end. Each subsystem was tested prior to three real-world tests: two at the Ocean Towing Tank at University College London (UCL) and one at Mercers Lake in Redhill, UK. Precautions were taken to mitigate the risk of drone loss or tank damage. The drone's sonar system ensured it remained clear of the

lake or tank floor, with a fail safe in the code programmed to prompt resurfacing in case of failure, allowing for easy retrieval. Additionally, a retrieval rope was attached to the drone as a precautionary measure. To prevent collision damage during UCL tests, the glider was padded with foam and a net was employed to protect against tank contact.

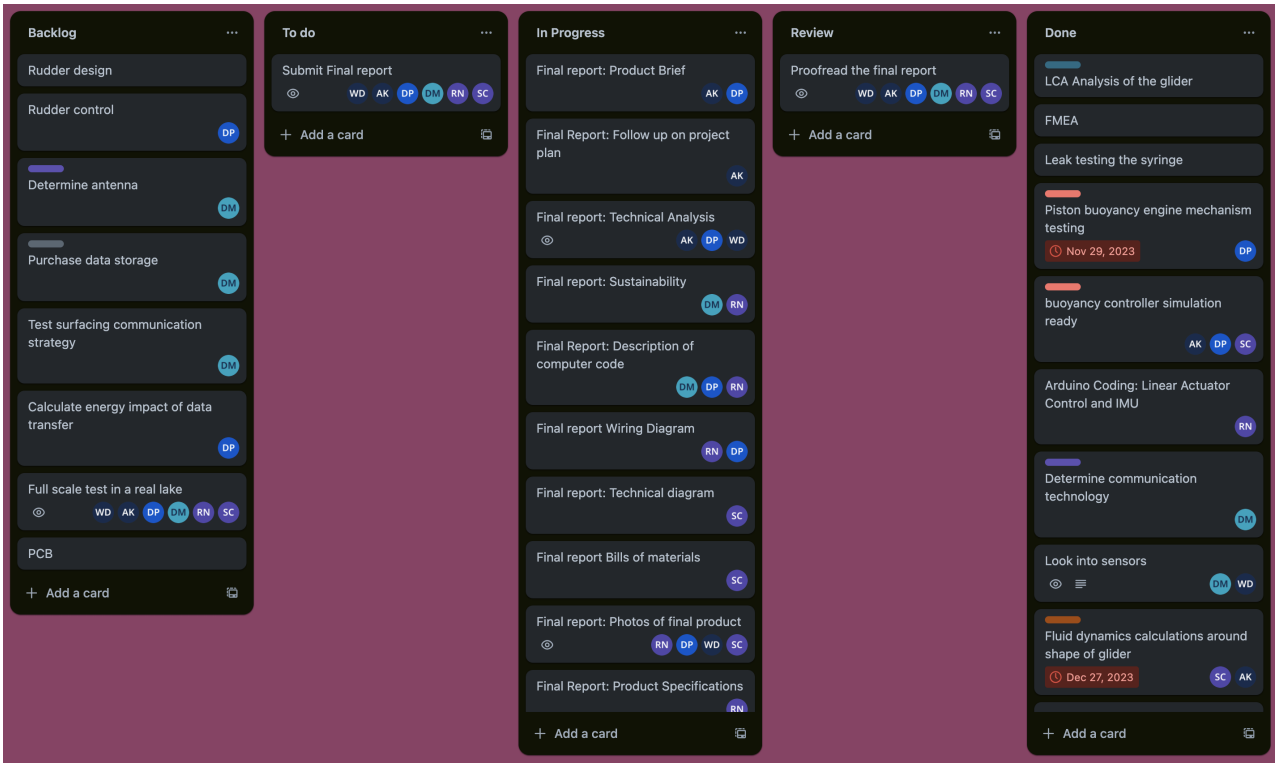


Figure 3: Final Trello board

Follow Up References

[1] J. Kang, F. Yan, P. Zhang, and C. Du, Comparison of comprehensive properties of Ni-MH (nickel-metal hydride) and Li-ion (lithium-ion) batteries in terms of energy efficiency *Energy*, vol. 70, pp. 618–625, 2014.

[2] M. A. P. Mahmud, N. Huda, S. H. Farjana, and C. Lang, Comparative Life Cycle Environmental Impact Analysis of Lithium-Ion (LiIo) and Nickel-Metal Hydride (NiMH) Batteries *Batteries*, vol. 5, no. 1, 2019.

[3] G. Majeau-Bettez, T. R. Hawkins, and A. H. Strømman, Life Cycle Environmental Assessment of Lithium-Ion and Nickel Metal Hydride Batteries for Plug-In Hybrid and Battery Electric Vehicles *Environmental Science & Technology*, vol. 45, no. 10, pp. 4548–4554, 2011. PMID: 21506538.

2.2 Technical Analyses

2.2.1 Buoyancy Engine

As discussed earlier, the only propulsion mechanism of this underwater drone is a buoyancy engine. After debating the different options, it was decided to go with a syringe based design for the buoyancy engine due to it being used in most state of the art underwater gliders [4]. The syringe module is composed of four main parts: the two syringes, a linear actuator, and a link between the syringes and the shaft of the actuator. The first step towards designing the buoyancy engine and selecting the correct parts was laying out the different constraints and requirements this engine has to meet:

- Size: A very strict size requirement of 13cm for the maximum diameter, flexible in terms of length
- Strength: Being able to withstand the 70 metres target depth
- Availability: Needed to be Purchasable from approved suppliers

After considering the constraints, the next step is to translate the constraints into specs for each section. There is no further analysis of the specs for the size, apart from making sure that the individual parts can be arranged in a suitable way to fit the glider. However, regarding the strengths of the parts for the target depth, a pressure and force analysis for the maximum depth is important to make sure that each part is up to specs. The pressure at 70 m is 684.44 kPa [5], so incorporating a safety margin, the target pressure rating is set at 800 kPa. The only type of syringe rated for this kind of pressure available from suppliers was car oil and brake fluid inspection syringes, as those circuits are under pressure so such syringes are rated for very high amounts of pressure (1200kPa). After selecting a type of syringe and deciding that 2 of them will be used, the force exerted on the pistons by the pressure of the water needs to be calculated to spec out the materials for the rest of the engine. The force is calculated using the following formula:

$$F_{\text{newton}} = P_{\text{pascal}} \cdot A_{m^2} \tag{1}$$

Depth	Pressure (kPa)	Force needed to overcome pressure (N)
0	0	0
10	97.78	283
20	195.55	567
30	293.33	850.657
40	391	1134
50	488.88	1417
60	586.66	1701
70	684.44	1984
80	782	2267.8
90	879.99	2551
100	977	2835

Table 2: Relation between Depth, Pressure and Force

The table 2.2.1 summarises the calculations based on a 4.3 cm diameter for each internal piston of the syringes, resulting in an area of 14.522 cm^2 per piston and a total area of 29 cm^2.

According to the calculations, the entire buoyancy engine assembly should be able to withstand 1984 N, therefore, the decision was made to aim for a 2500 N rating for the entire assembly. To calculate the space left after adding the syringes:

$$\text{space available} = \text{tube area} - \text{area taken by syringe} \cdot 2 = 132 - 33 = 99\text{cm}^2 \tag{2}$$

Considering the size constraints and the force required, a linear actuator with a base of 4.2x6 cm ($30cm^2$) and a rated strength of 2500 N was selected. In addition to checking the theoretical space taken by the linear actuator in the tube, both syringes and linear actuator were fully modelled in CAD to check that all the components could be arranged to fit inside the hull.

The last part to design was the link between the syringes and the linear actuator (page 29). The material needed to be easily manufacturable, inexpensive, and strong enough for the application. After careful consideration of different materials (aluminium, wood, acrylic, 3D printed PLA, ABS...) and research papers assessing their respective strength, 3D printed PLA seemed like the best option:

- Easily manufacturable, therefore, iterating and adjusting the design is relatively effortless
- Readily available and inexpensive to manufacture

- Strong enough based on multiple research papers [6] it has a resistance of 35-50 mPa, with print parameters of 0.05 - 0.2 layer height and 100% infill, for an evenly distributed force across the part.

Even if load is not distributed across the entire part, the target pressure is 0.85 mPa, therefore, with such a large margin of error, there was confidence that the part would hold. Future steps towards ensuring the strengths of the part would be to do a full Finite Element Analysis (FEA) with directional loads and real-world stress testings.

To hold the linear actuator and the syringes together, a two-part mount was designed. The initial part of this assembly (on page 27) holds both syringes and actuators with a chemical bond by epoxy resin. Epoxy resin was chosen due to its ability to bond heterogeneous materials together very well, with a shear resistance of $5mPa$ to $97mPa$ (depending on the materials) [7] and was available from suppliers. The second part of the assembly has the purpose of holding the base of the actuator with a metal dowel (page 28). Both parts are screwed to the inside mesh structure of the glider to be held as one module (as seen on page 19).

2.2.2 Battery Module

The battery module was the most intricate part of the glider to design, thanks to its weight, size and dual-functionality. The first responsibility of the battery is to power the electronics and the second is to move the center of mass by being mounted on a linear actuator. All the major components of the drone, the linear actuators and the Arduino, can run on 12V, the Arduino can step down the voltage to 3.3V and 5V for sensors that require it. For more flexibility, the battery pack was built from 10 individual cells of 1.2V each and arranged around the linear actuator in a pentagonal shape (page 20), therefore, the shaft of the linear actuator also acts as a guide rail for the movement of the battery 14. In addition, having the battery around the liner actuator and not mounted at the end of it allows to save 16cm in length. The following formula is used to calculate the impact of the battery movement on the centre of mass [8]:

$$\Delta CM = \frac{m \cdot d}{M} \tag{3}$$

Knowing that the total mass of the glider is 31.93kg, the weight of the battery is 4.37kg and the linear actuator travels for 30cm:

$$\Delta CM = \frac{4.37 \cdot 0.3}{31.93} = 0.041m = 4.1cm \tag{4}$$

The battery module on its own can move the centre of mass by 4.1cm, however, this is not enough to create the difference in angles as observed in figure 4, because, in addition to the battery module, the syringe module has an intended secondary effect of moving the centre of mass by filling up with water. In order to calculate the effect of this change of mass at the very tip of the glider, the following formula can be used [8]:

$$COM_{\text{new}} = \frac{m_{glider} \cdot x_{cm} + m_{water} \cdot x_{syringe}}{m_{glider} + m_{water}} \tag{5}$$

Knowing that the centre of mass of the glider is 101.23cm away from the tip of the glider, calculated by finding the tipping point of the glider on a flat and levelled surface, the middle of the syringes (middle point of the variable mass) is located 12.4cm away from the tip and the total capacity of the syringes is 400ml, therefore, a variable mass of 0g to 400g:

$$\Delta COM = \frac{31.93 \cdot 101.23 + 0.40 \cdot 8.17}{31.93 + 0.40} = 100.05 \tag{6}$$

$$\Delta COM = COM_{\text{original}} - COM_{\text{new}} = 101.23 - 100.05 = 1,18cm \tag{7}$$

So the compounded effect of the battery module and the syringe module on the centre of mass is $1.18 + 4.1 = 5.28cm$. However, this number seems too low to explain the drastic change in the angle

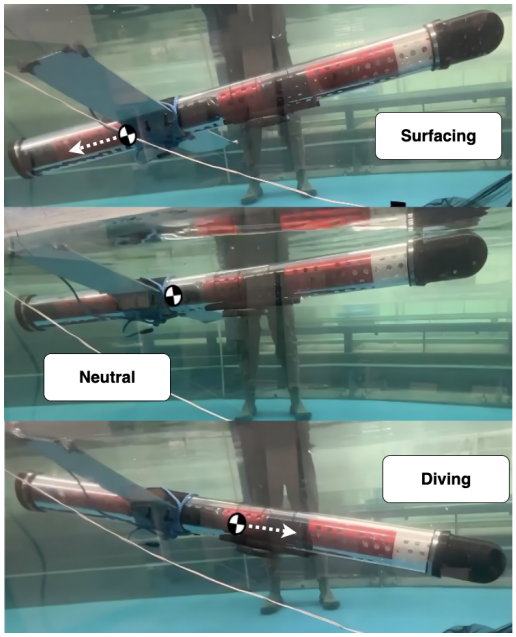


Figure 4: Centre of mass during the 3 possible states of the glider

between the Surfacing and Diving motions of the glider, the explanation for that is that the syringes also modify the centre of buoyancy, which is dissociated from the centre of mass (similar to the centre of mass and centre of lift in planes), and this change in buoyancy balance is suspected to be the additional force behind the angle change. Unfortunately, the centre of buoyancy is much more difficult to estimate with complex internal geometry and might require simulation tools or real-world tests.

In conclusion, the exact impact of the change in the centre of buoyancy is not required for the pitch control to work, the battery does not have a significant enough impact on the centre of mass to create these angles on its own, but is sufficient to influence the glider's angle of attack to achieve the optimal angle, which is of most importance.

2.2.3 Wings

The design of the wings plays a crucial role in determining the glider's performance and efficiency. This section delves into the mathematical calculations used to derive the wing design parameters, drawing inspiration from the sweep wing strategy as established in the paper by [9]. The sweep wing

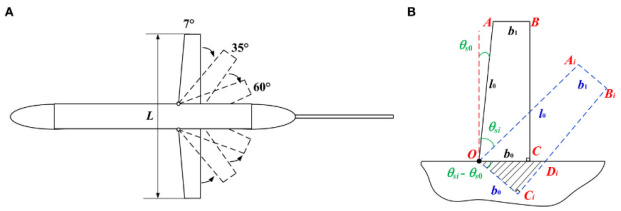


Figure 5: Wing sweep angles and parameters. From Wang, Y., Wang, C., Yang, M., Liang, Y., Han, W., Yang, S. (2022). Glide performance analysis of underwater glider with sweep wings inspired by swift. Frontiers in Marine Science, 9, 1048328.

strategy is illustrated in Figure 5 which was taken from [9]. The wings are articulated with the body of the underwater glider (UG), where L represents the wing span. Figure 5B, also taken from the paper, depicts the relationship between the wing area and the aspect ratio with the sweep angle θ_s, which can vary from 7° to 60°.

The area of a single wing, denoted as S_{wing}, is described geometrically as rectangle $OABCD$ at an arbitrary sweep angle θ_{si}. This area can be mathematically calculated using the following formula:

$$S_{\text{wing}} = (b_1 + b_0)l_0 \cos(\theta_{s0}) - b_0^2 \tan(\theta_{si} - \theta_{s0}) \tag{8}$$

Here, b_0 and b_1 represent the chord of the wing root and tip, respectively, and l_0 is the length of the wing leading edge.

Furthermore, the aspect ratio λ of the wings is a critical parameter in aerodynamic design. It can be calculated using the following formula:

$$\lambda = \frac{((2l_0 \cos(\theta_{si}) + D))^2}{2S_{\text{wing}}^2} \tag{9}$$

Where D represents the glider's diameter.

The versatility of these equations allows for the design of wings with varying mission-specific sweep angles. While the initial iteration of the glider employed a 0-degree sweep angle, the modular design of the glider facilitates the attachment of wings with different sweep angles to the hull without necessitating a complete redesign.

The aspect ratio for the glider is calculated to be 3, significantly enhancing its maneuverability and aerodynamic efficiency. A low aspect ratio ensures high maneuverability due to a lower moment of inertia, as discussed in [10]. This characteristic is particularly crucial for navigating confined spaces such as the underwater environment of a lake's surface. These rigorous mathematical calculations and the adaptable design approach contribute significantly to optimising the wing configuration of autonomous underwater gliders, enhancing their operational efficiency and adaptability across diverse environmental conditions.

2.2.4 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) simulations play a crucial role in providing comprehensive insights into the flow patterns and pressure distribution surrounding the hull and wings of a glider. This computational analysis aids in the iterative optimisation of the glider's design, leading to enhanced performance and efficiency. For the purpose of this project, boundary conditions such as flow velocity, temperature, and turbulence model were meticulously set to emulate the underwater lake environment accurately. The analysis of pressure distribution over the hull and all other sur-

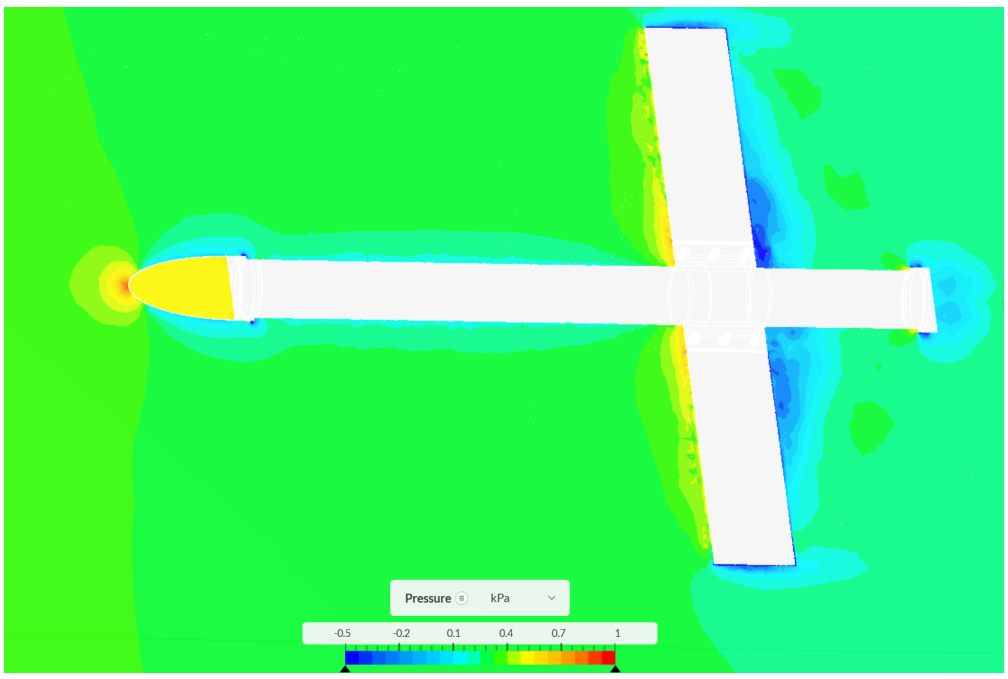


Figure 6: Pressure Distribution around the glider

faces unveiled critical areas of potential flow separation, boundary layer effects, and regions prone

to heightened drag. Through a thorough examination of these pressure gradients, strategic modifications were incorporated to minimise drag, improve stability, and consequently optimise energy consumption. The pressure distribution around the glider is visually depicted in Figure 6, showcasing the pressure gradients. Subsequently, leveraging the finalised glider design, rigorous calculations

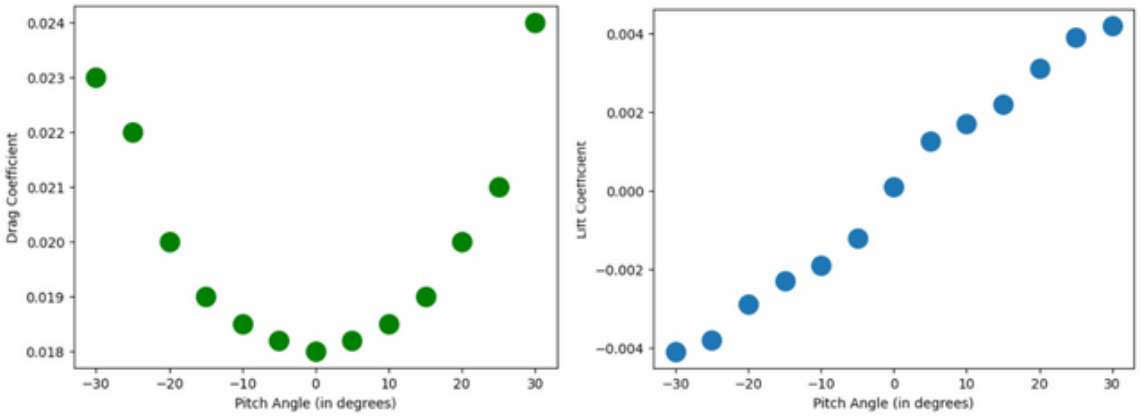


Figure 7: Lift Coefficient and Drag Coefficient as functions of Pitch Angle

were conducted to ascertain the lift and drag coefficients. These coefficients serve as fundamental parameters in evaluating the hydrodynamic performance and behavior of the glider under varying conditions. To further explain the hydrodynamic characteristics of the glider, Figure 7 illustrates the Lift Coefficient and Drag Coefficient as functions of the Pitch Angle, providing valuable insights into the glider's hydrodynamic efficiency and stability profile.

2.2.5 Dynamics and Control

The glider is passively propelled i.e there is no propulsion system that actively utilises energy to propel the glider forward. The motion of the glider in the vertical plane is solely controlled by the buoyant and gravitational forces acting upon it with an internal moving mass used to fine tune its pitch angle [11]. The net buoyancy force determines the direction of motion with the glider moving up when net buoyancy is positive and moving down when the net buoyancy is negative. The upward buoyant force on the glider is given by

$$F_{\text{buoyancy}} = \rho_{\text{water}} \cdot V_{\text{displaced}} \cdot g \tag{10}$$

Where:

$$\begin{aligned} \rho_{\text{water}} &: \text{Density of water} \\ V_{\text{displaced}} &: \text{Volume of water displaced during ascent} \\ g &: \text{acceleration due to gravity} \end{aligned}$$

The downward force due to the glider's weight is given by:

$$F_{\text{weight}} = M \cdot g \tag{11}$$

Where:

$$\begin{aligned} M &: \text{Mass of the glider} \\ g &: \text{acceleration due to gravity} \end{aligned}$$

$$F_{\text{net}} = F_{\text{buoyancy}} - F_{\text{weight}} \tag{12}$$

Another force that affects the motion of the glider is drag which depends upon the drag coefficient of the glider, the area and its gliding speed. The equation for drag force (F_{drag}) is given by:

$$F_{\text{drag}} = \frac{1}{2}\rho v^2 C_d A \tag{13}$$

Where:

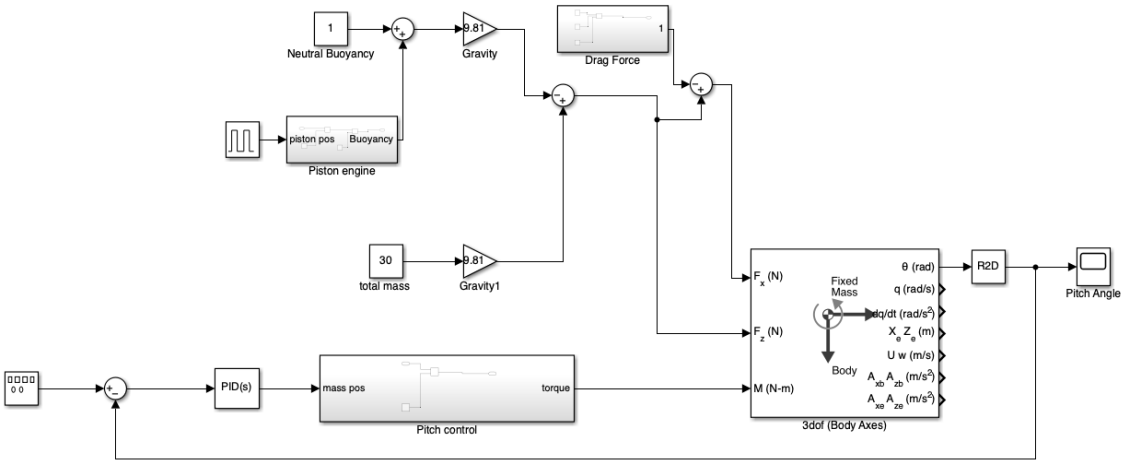


Figure 8: SIMULINK Model of Glider

- ρ is the density,
- v is the velocity of the glider
- C_d is the drag coefficient,
- A is the reference area of the glider

Equations (1) - (4) along with information obtained from [11], [12] and [13] were used to generate a SIMULINK model of the glider as can be seen in fig.8. and obtain some basic insights into other aspects of the glider such as the weight required to achieve neutral buoyancy.

The final specifications of the glider were obtained through simulation and experimentation. They can be seen in table 2.2.5

Table 3: Technical Specifications

Specification	**Value**
Mass for Neutral Buoyancy	31.934 kg
Ballast Mass	400 g
Drag Coefficient	0.018
Movable Mass	4.37kg
Pitch Angle	30°
Initial Position of Movable Mass	35cm from Tail
Speed of Glider	30cm/s
K_p	0.003749
K_i	0.000419
K_d	0.007973

Analysis References

[4] B. Page, S. Ziaeefard, A. Pinar, and N. Mahmoudian, Highly Maneuverable Low-Cost Underwater Glider: Design and Development *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, 10 2016.

[5] "Depth-to-pressure calculator." https://bluerobotics.com/learn/pressure-depth-calculator/, Dec 2021.

[6] W. Wang, B. Zhang, L. Zhao, M. Li, Y. Han, L. Wang, Z. Zhang, J. Li, C. Zhou, and L. Liu, Fabrication and properties of PLA/nano-HA composite scaffolds with balanced mechanical properties and biological functions for bone tissue engineering application *Nanotechnology Reviews*, vol. 10, pp. 1359–1373, 10 2021.

[7] "Material property data." https://www.matweb.com/search/datasheet_print.aspx?matguid=956da5edc80f4c62a72c15ca2b923494, 2024. Accessed: 2024-04-14.

[8] D. Halliday, R. Resnick, and J. Walker, *Fundamentals of Physics*. Hoboken, NJ: Wiley, 11th ed., 2020.

[9] Y. Wang, C. Wang, M. Yang, Y. Liang, W. Han, and S. Yang, Glide performance analysis of underwater glider with sweep wings inspired by swift *Frontiers in Marine Science*, vol. 9, p. 1048328, 2022.

[10] Y. Li, D. Pan, Z. Ma, and Q. Zhao, Aspect ratio effect of a pair of flapping wings on the propulsion of a bionic autonomous underwater glider *Journal of Bionic Engineering*, vol. 16, pp. 145–153, 2019.

[11] J. Huang, H.-S. Choi, D.-W. Jung, J.-H. Lee, M.-J. Kim, K.-B. Choo, H.-J. Cho, and H.-S. Jin, Design and motion simulation of an underwater glider in the vertical plane *Applied Sciences*, vol. 11, no. 17, p. 8212, 2021.

[12] D. Grande, L. Huang, C. A. Harris, P. Wu, G. Thomas, and E. Anderlini, Open-source simulation of underwater gliders in *OCEANS 2021: San Diego–Porto*, pp. 1–8, IEEE, 2021.

[13] J. G. Graver, *Underwater gliders: Dynamics, control and design*. PhD thesis, Princeton University Princeton, 2005.

2.3 Stakeholder and Sustainability Analysis

2.3.1 Sustainability Analysis

Autonomous underwater gliders can play a crucial role in climate change research by gathering real-time data on essential environmental factors like temperature, salinity, and dissolved oxygen levels. This technology presents a promising avenue with distinct advantages related to sustainability. These aspects have been thoroughly analysed in accordance with the United Nations Sustainable Development Goals (SDGs) set forth in Agenda 2030 [14].

The glider can be modified to collect real-time data on critical environmental parameters such as temperature, salinity, and dissolved oxygen levels. This data is essential for understanding climate-related phenomena and developing effective mitigation and adaptation strategies, directly contributing to SDG 13: Climate Action [15]. Moreover, the continuous monitoring capabilities of underwater gliders enable the assessment of water quality in lakes and aquatic ecosystems. They detect pollutants, algal blooms, and other water quality indicators, facilitating sustainable water management practices and aligning with SDG 6: Clean Water and Sanitation [16]. Additionally, the deployment and utilisation of underwater gliders represent advancements in science, technology, and innovation. These advancements contribute to SDG 9: Industry, Innovation, and Infrastructure [17] by fostering technological innovation in environmental monitoring and conservation efforts.

To ensure sustainability, particular attention is given to the environmental impact of underwater gliders. Efforts are made to enhance their energy efficiency and minimise their carbon footprint during operations. Rigorous evaluations are conducted to assess their impact on marine ecosystems and water quality, with a focus on the sustainability of materials used in their construction. Furthermore, the choice of NiMH batteries in gliders reflects a commitment to sustainability. These batteries

have been found to have the least environmental impact according to the Ecolizer 2.0 Life Cycle Assessment (LCA) tables [18], highlighting a dedication to responsible environmental stewardship in technology development.

2.3.2 Ethical and Security Considerations

The deployment of autonomous underwater gliders raises significant ethical concerns that demand careful deliberation and proactive measures to mitigate potential adverse impacts. In this section, these ethical considerations as moral dilemmas are discussed and strategic approaches to address them effectively are proposed.

Balancing Data Collection with Ecological Disruption

Factors in Favor of Data Collection

- Enhanced understanding of the lake ecosystem facilitates the formulation of effective management strategies for conservation and resource protection.
- Glider-collected data plays a pivotal role in early detection of environmental threats, enabling timely intervention to mitigate potential risks.

Ecological Disruption Factors:

- There is a risk of inadvertent collisions or harm to aquatic organisms during glider operation, highlighting potential ethical implications.

Strategies to Address the Dilemma:

- *Zoning Regulations:* Establishing designated operating zones within the lake, taking into account critical habitat areas and seasonal migrations of wildlife, can minimise direct interference with sensitive ecosystems.
- *Real-time Monitoring and Adaptation:* Leveraging real-time data analysis capabilities allows for the identification of potential interactions with wildlife. This enables adaptive management practices, where glider operations can be adjusted promptly to minimise ecological impact.

Open Access to Data vs. Proprietary Information

Factors in Favor of Open Access:

- Openly sharing glider-collected data with the scientific community promotes collaboration and accelerates research efforts aimed at enhancing lake health.
- Public access to data empowers stakeholders, including policymakers and local communities, to make informed decisions regarding lake management and conservation efforts.

Factors Supporting Proprietary Information:

- Considerable investment is often required for the development and deployment of glider technology, necessitating a return on investment through data commercialisation.
- Sensitive or proprietary data, such as real-time monitoring results or specific location information, may require protection to prevent unauthorised use or misinterpretation.

Strategies for Balancing Open Access and Proprietary Information:

- *Tiered Data Access System:* Implementing a tiered data access model can strike a balance between open access and proprietary needs. Basic datasets can be made publicly available, while more detailed analysis or real-time data may require a licensing or subscription model.
- *Clear Data-Sharing Policies:* Developing robust data-sharing policies is essential to ensure scientific integrity and prevent unauthorised use. These policies should outline guidelines for data access, usage, and intellectual property rights, fostering responsible data sharing and collaboration among researchers.

2.3.3 Stakeholder Analysis

Understanding the interests, concerns, and roles of stakeholders is crucial for ensuring the success, ethical integrity, and sustainability of innovative technologies such as the autonomous underwater glider. Through a thorough analysis of the perspectives and expectations of various stakeholders, key priorities can be identified and potential challenges addressed. These efforts can contribute significantly to the responsible and effective utilisation of the autonomous underwater glider in lake monitoring and research initiatives, thereby enhancing its overall impact and relevance in the field. Table 4 provides an overview of the diverse stakeholder groups involved. It offers insights into the potential interests and concerns that different groups may possess regarding the autonomous underwater glider.

Table 4: Stakeholder groups and their potential interests & concerns

Stakeholder Group	**Interests**	**Concerns**
Researchers and Scientists	Access to high-quality environmental data, technological advancements for research, collaboration opportunities	Data accuracy, reliability of glider systems, data accessibility, collaboration frameworks
Environmental Agencies and NGOs	Reliable data for environmental monitoring, policy support based on scientific evidence, sustainable resource management	Data transparency, ethical data use, alignment with conservation goals, community engagement
Government and Regulatory Bodies	Regulatory compliance, informed decision-making, resource allocation for environmental initiatives, public safety	Data security, privacy, regulatory standards adherence, public trust and confidence
Technology Developers and Manufacturers	Technological innovation, market demand, product performance, industry partnerships	R&D investment, product reliability, intellectual property protection, market competition
Local Communities and Stakeholder Groups	Environmental impact mitigation, community involvement, public awareness, sustainable practices	Environmental disruption, community consultation, data governance, socio-economic impacts
Commercial Entities and Industry Partners	Data monetization, commercial applications, market expansion, investment opportunities	Data ownership, licensing agreements, market volatility, ethical business practices

Sustainability References

[14] "THE 17 GOALS — Sustainable Development — sdgs.un.org." https://sdgs.un.org/goals. [Accessed 14-04-2024].

[15] "Goal 13 — Department of Economic and Social Affairs — sdgs.un.org." https://sdgs.un.org/goals/goal13. [Accessed 14-04-2024].

[16] "Goal 6 — Department of Economic and Social Affairs — sdgs.un.org." https://sdgs.un.org/goals/goal6. [Accessed 14-04-2024].

[17] "Goal 9 — Department of Economic and Social Affairs — sdgs.un.org." https://sdgs.un.org/goals/goal9. [Accessed 14-04-2024].

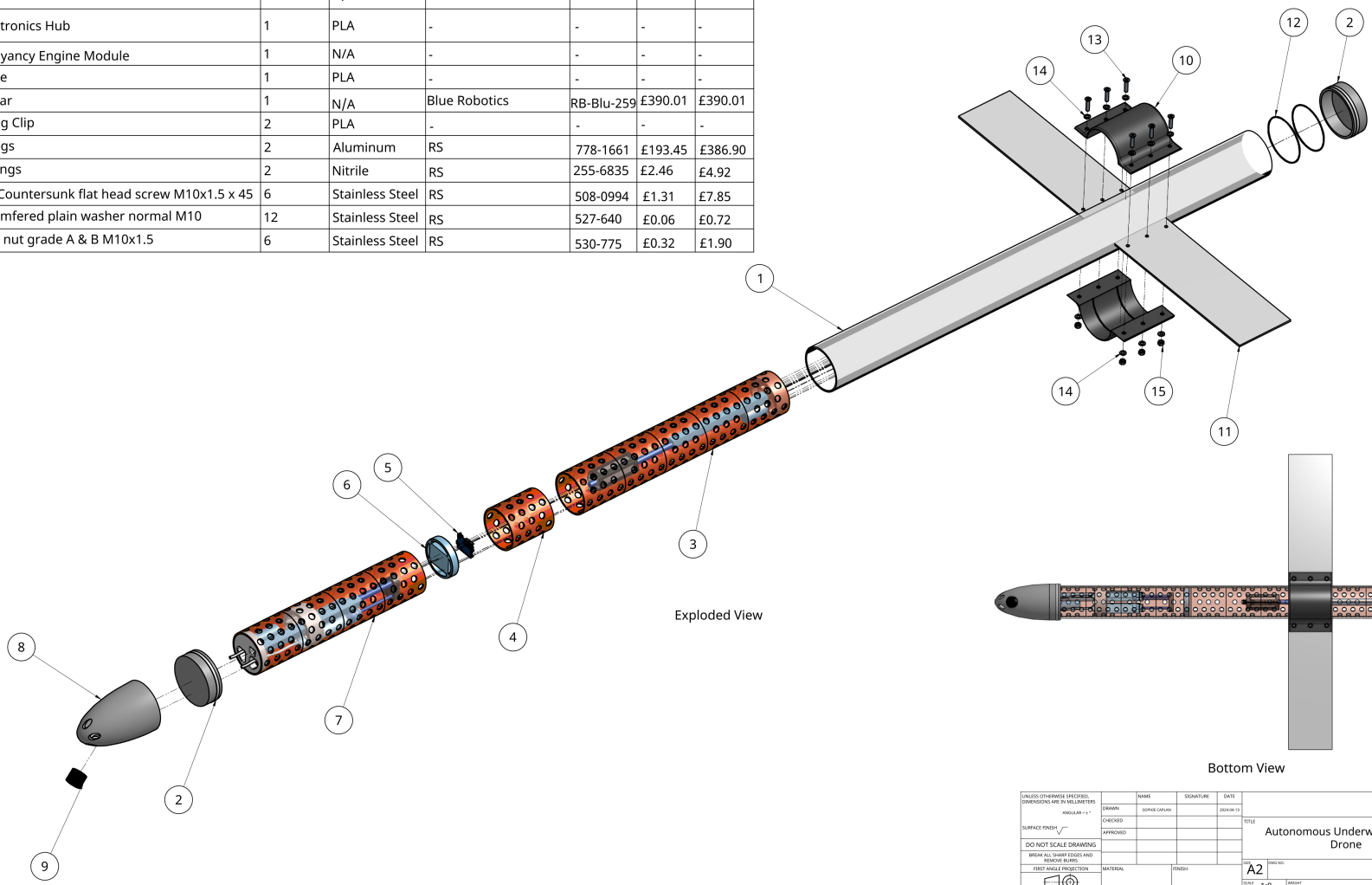
[18] "Dienstverlening — ovam.vlaanderen.be." https://ovam.vlaanderen.be/. [Accessed 14-04-2024].

2.4 Technical Diagrams

The engineering drawings for the final glider and all its components are included. Parts are referenced on the Bill of Materials which is further detailed in 2.5. Note that adjustments were made to design revision F from the glider that was tested as described in 2.8. These are minor aesthetic adjustments, such as the design of the nose.

Exploded assembly drawings have been provided which detail each component in the glider. Individual drawings are provided for the components which were designed to be manufactured.

Item	Name	Quantity	Material	Vendor	Stock number	Unit Cost	Cost
1	Hull	1	Acrylic	The Clear Plastic Shop	XT	£220.10	£220.10
2	End Cap	2	N/A	Toolstation	78470	£14.68	£29.36
3	Battery Module	1	N/A	-	-	-	-
4	Mesh	1	PLA	-	-	-	-
5	Arduino Uno	1	N/A	RS	266-2937	£25.74	£25.74
6	Electronics Hub	1	PLA	-	-	-	-
7	Buoyancy Engine Module	1	N/A	-	-	-	-
8	Nose	1	PLA	-	-	-	-
9	Sonar	1	N/A	Blue Robotics	RB-Blu-259	£390.01	£390.01
10	Wing Clip	2	PLA	-	-	-	-
11	Wings	2	Aluminum	RS	778-1661	£193.45	£386.90
12	O-rings	2	Nitrile	RS	255-6835	£2.46	£4.92
13	PH Countersunk flat head screw M10x1.5 x 45	6	Stainless Steel	RS	508-0994	£1.31	£7.85
14	Chamfered plain washer normal M10	12	Stainless Steel	RS	527-640	£0.06	£0.72
15	Hex nut grade A & B M10x1.5	6	Stainless Steel	RS	530-775	£0.32	£1.90



Exploded View

Bottom View

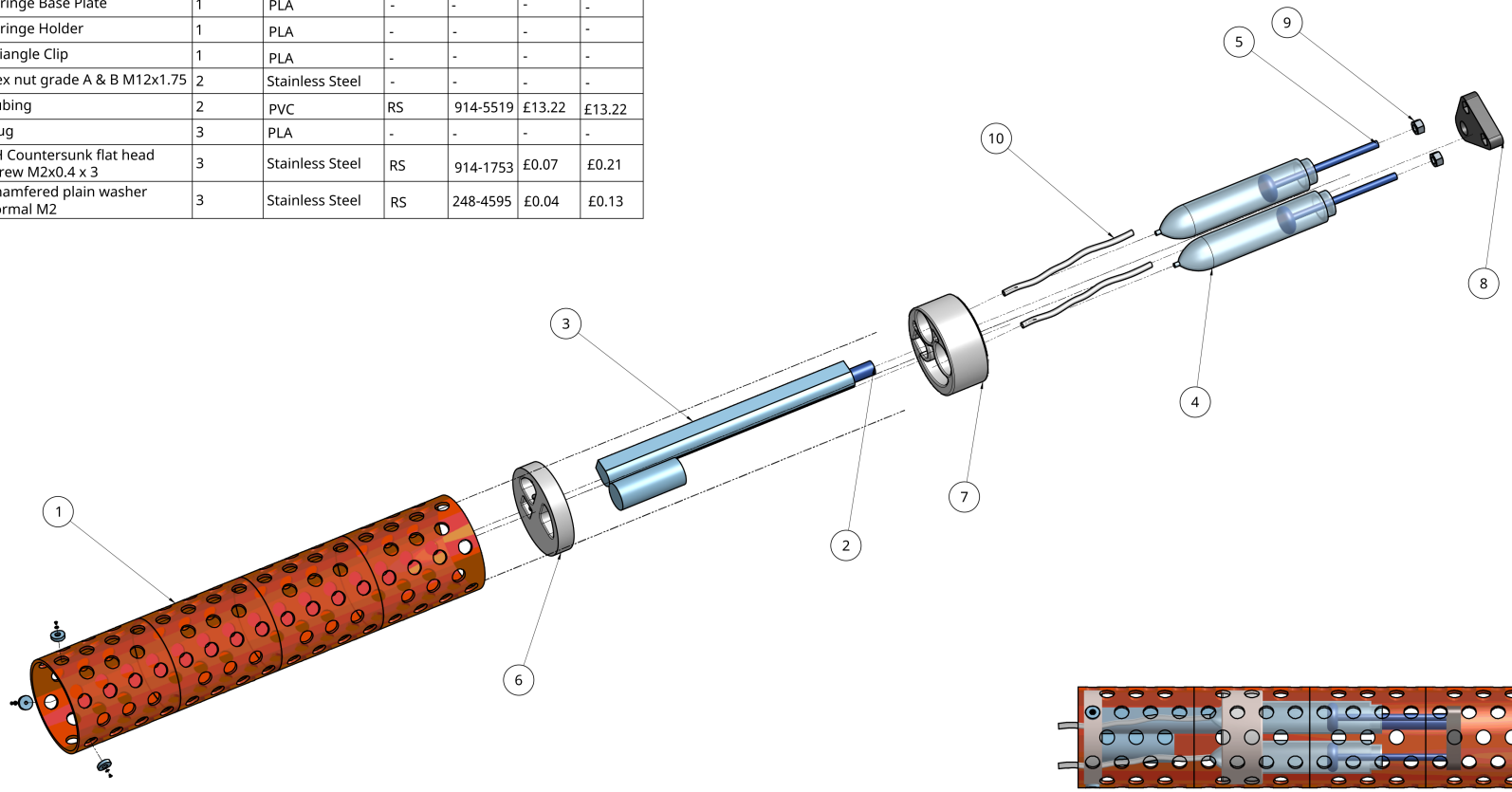
		NAME	SIGNATURE	DATE
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS	DRAWN	SOPHIE CAPLAN		2024-04-19
ANGULAR: ± °	CHECKED			
SURFACE FINISH:	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL		FINISH	

TITLE: Autonomous Underwater Glider Drone

SIZE: A2 | DWG NO. | REV: F

SCALE: 1:8 | WEIGHT | SHEET: 1 of 1

Item	Name	Quantity	Material	Vendor	Stock number	Unit Cost	Cost
1	Mesh	4	PLA	-	-	-	-
2	Linear Actuator Shaft	1	-	RS	177-4510	-	-
3	Linear Actuator	1	-	RS	177-4510	£344.98	£344.98
4	Syringe	2	-	Rapid	89-8733	£25.09	£50.18
5	Syringe Plunger	2	-	Rapid	89-8733	-	-
6	Syringe Base Plate	1	PLA	-	-	-	-
7	Syringe Holder	1	PLA	-	-	-	-
8	Triangle Clip	1	PLA	-	-	-	-
9	Hex nut grade A & B M12x1.75	2	Stainless Steel	-	-	-	-
10	Tubing	2	PVC	RS	914-5519	£13.22	£13.22
11	Plug	3	PLA	-	-	-	-
12	PH Countersunk flat head screw M2x0.4 x 3	3	Stainless Steel	RS	914-1753	£0.07	£0.21
13	Chamfered plain washer normal M2	3	Stainless Steel	RS	248-4595	£0.04	£0.13



Right Side View

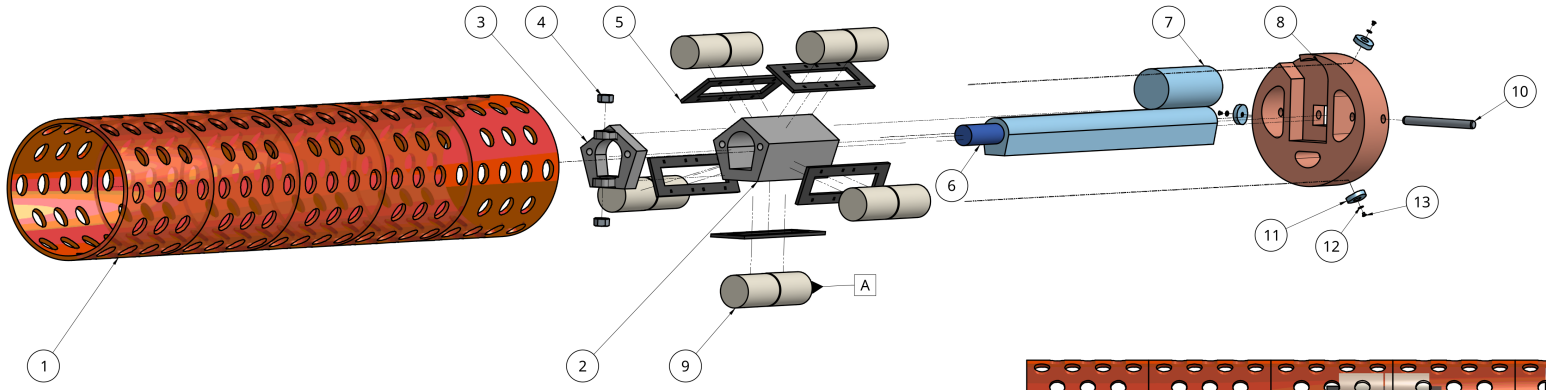
		NAME	SIGNATURE	DATE
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS	DRAWN	SOPHIE CAPLAN		2024-04-19
ANGULAR: ± °	CHECKED			
SURFACE FINISH:	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL		FINISH	

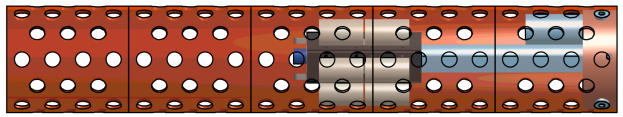
TITLE: Buoyancy Engine Module

SIZE: A2 | DWG NO. | REV: E

SCALE: 1:4 | WEIGHT | SHEET: 1 of 1

Item	Name	Quantity	Material	Vendor	Part number	Unit cost	Cost
1	Mesh	5	PLA	-	-	-	-
2	Pentagonal Battery Pack	1	PLA	-	-	-	-
3	Pentagonal Battery Pack Clip	1	PLA	-	-	-	-
4	Hex nut grade A & B M8x1.25	2	Stainless Steel	RS	530-769	£0.19	£0.37
5	Battery Plates	5	Acrylic	-	-	-	-
6	Linear Actuator Shaft	1	N/A	RS	177-4510	-	-
7	Linear Actuator	1	N/A	RS	177-4510	£344.98	£344.98
8	Battery Actuator Holder	1	PLA	-	-	-	-
9	D Batteries	10	N/A	Rapid	18-2512	£12.36	£123.60
10	Parallel pin 8 x 80	1	Stainless Steel	RS	270-552	£0.24	£0.24
11	Plug	3	PLA	-	-	-	-
12	Chamfered plain washer normal M2	3	Stainless Steel	RS	527-347	£0.02	£0.06
13	PH Countersunk flat head screw M2x0.4 x 3	3	Stainless Steel	RS	914-1753	£0.07	£0.21





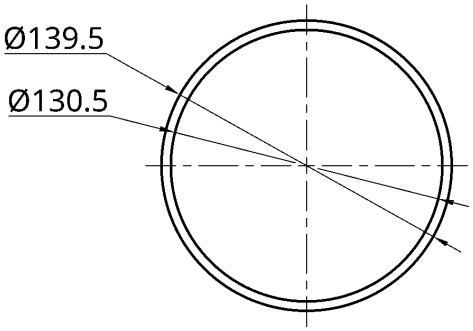
1:4

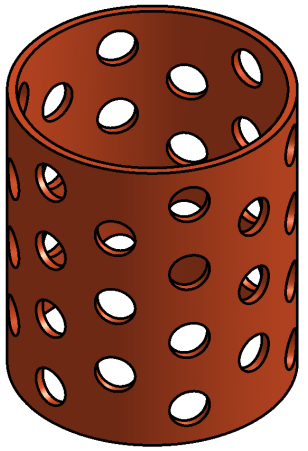
NOTES
A. Cable ties thread through holes in Item 6 and around batteries to secure them in place

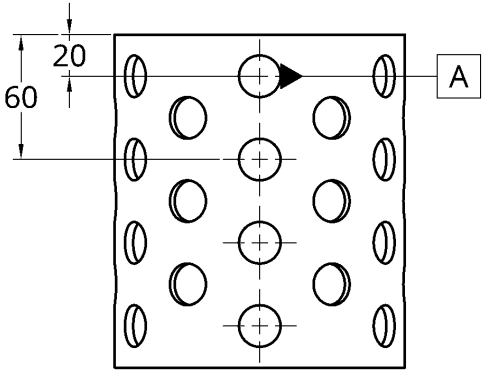
	NAME	SIGNATURE	DATE
DRAWN	SOPHIE CAPLAN		2024-04-14
CHECKED			
APPROVED			

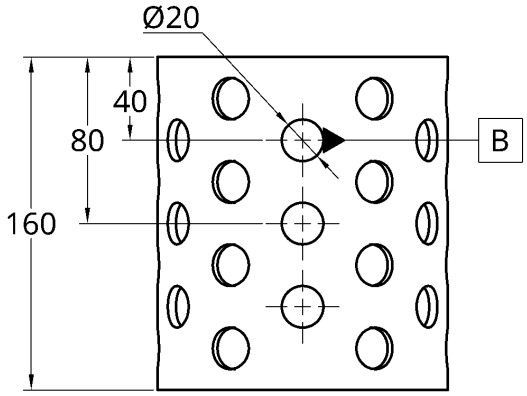
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS
ANGULAR = ± °
SURFACE FINISH
DO NOT SCALE DRAWING
BREAK ALL SHARP EDGES AND REMOVE BURRS
FIRST ANGLE PROJECTION
MATERIAL
FINISH

TITLE: Battery Module
SIZE: A2
DWG NO.
REV: F
SCALE: 1:3
WEIGHT
SHEET: 1 of 1







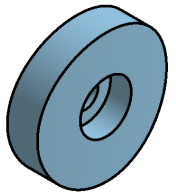


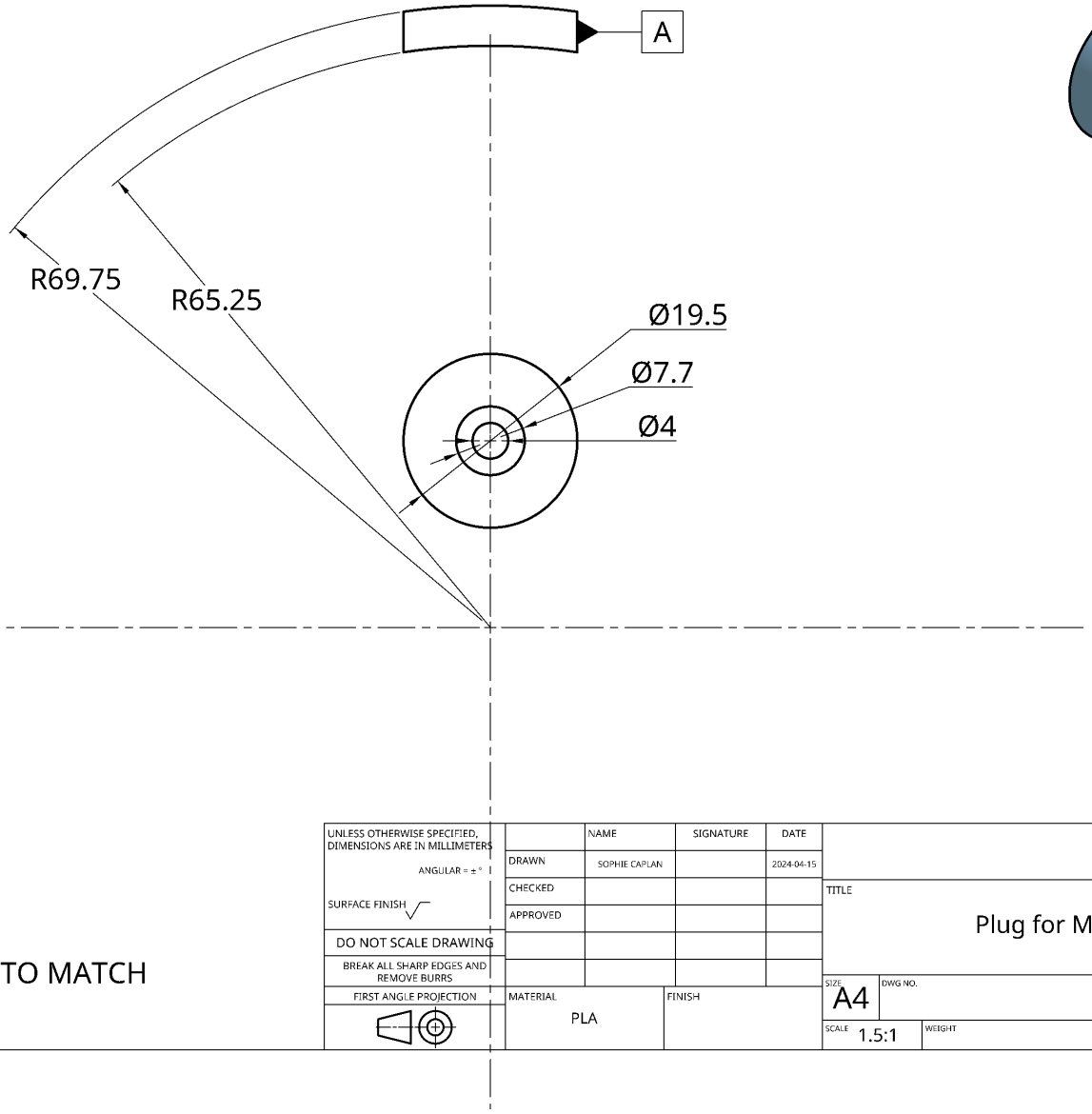
NOTES.
PATTERNS A AND B ARE REPEATED
4 TIMES EACH CIRCULARLY WITH
EVEN SPACING

UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS		NAME	SIGNATURE	DATE
ANGULAR = ± °	DRAWN	SOPHIE CAPLAN		2024-04-15
SURFACE FINISH	CHECKED			
	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL: PLA		FINISH:	

TITLE: Mesh

SIZE: A4	DWG NO.	REV.: A
SCALE: 1:3	WEIGHT:	SHEET: 1 of 1



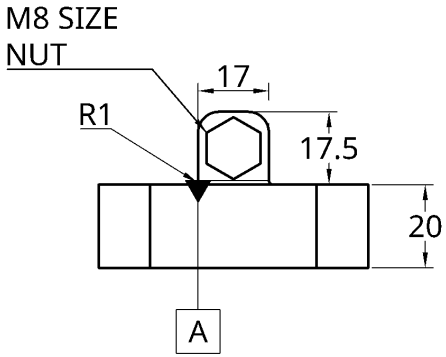


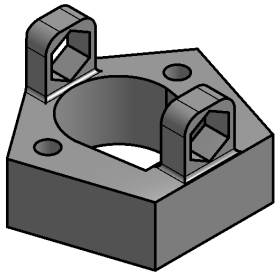
NOTES
A. CURVATURE OF PLUG TO MATCH MESH PERFECTLY FLUSH

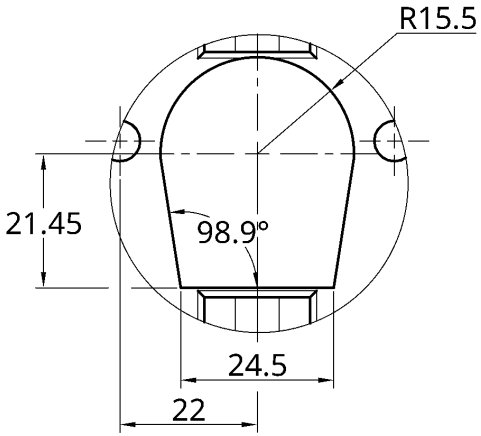
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS		NAME	SIGNATURE	DATE
ANGULAR = ± °	DRAWN	SOPHIE CAPLAN		2024-04-15
SURFACE FINISH	CHECKED			
	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL: PLA		FINISH:	

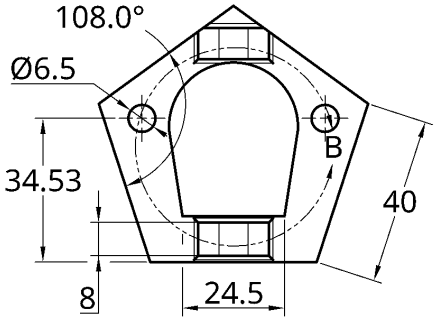
TITLE: Plug for Mesh

SIZE: A4	DWG NO.	REV.: A
SCALE: 1.5:1	WEIGHT:	SHEET: 1 of 1









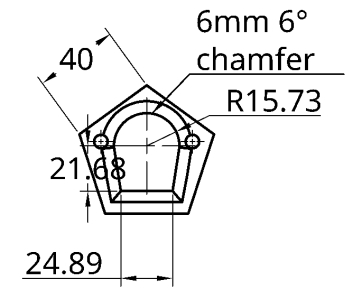
NOTES
A. FILLET EDGE ALL AROUND

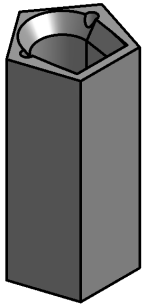
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS		NAME	SIGNATURE	DATE
ANGULAR = ± °	DRAWN	SOPHIE CAPLAN		2024-04-15
SURFACE FINISH	CHECKED			
	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL	PLA	FINISH	

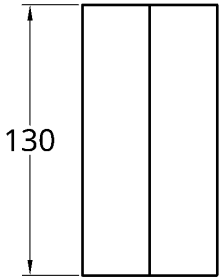
TITLE: Pentagonal Battery Shaft Top

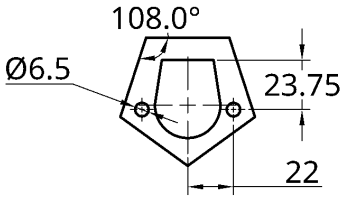
SIZE	DWG NO.	REV.
A4		A

SCALE	WEIGHT	SHEET
1:1.5		1 of 1









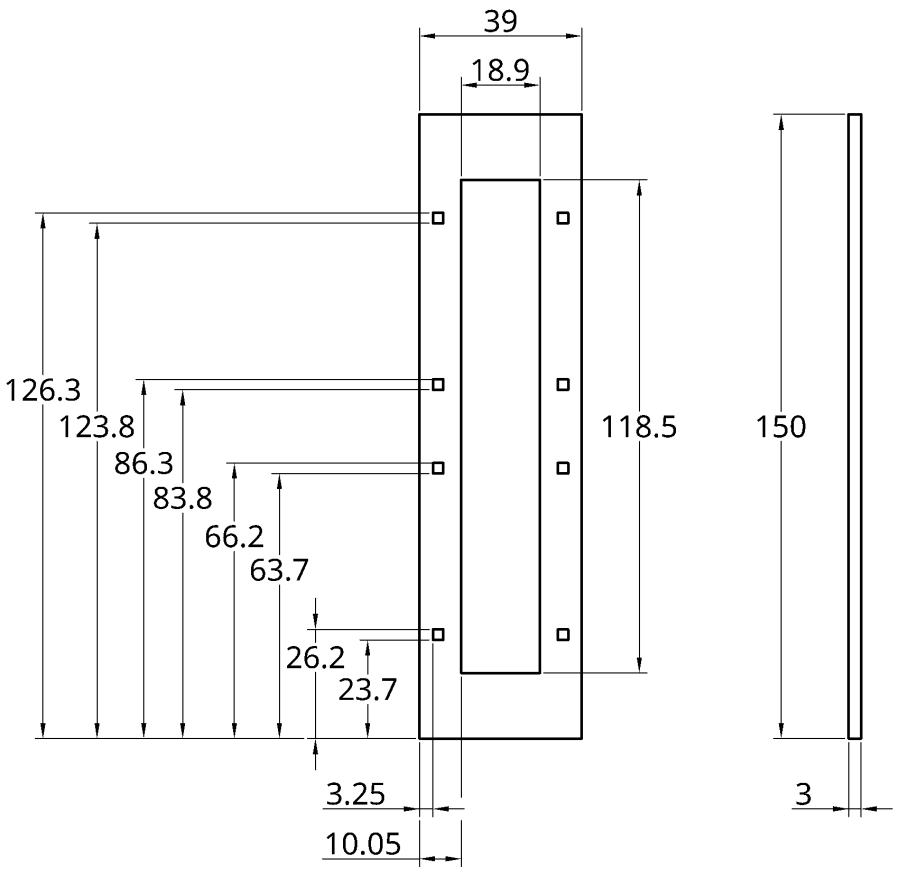
NOTES.
FLUSH END SECURED TO FLUSH END OF PENTAGONAL BATTERY SHAFT TOP.
DESGINED IN 2 PARTS FOR EASE OF ADDITIVE MANUFACTURING

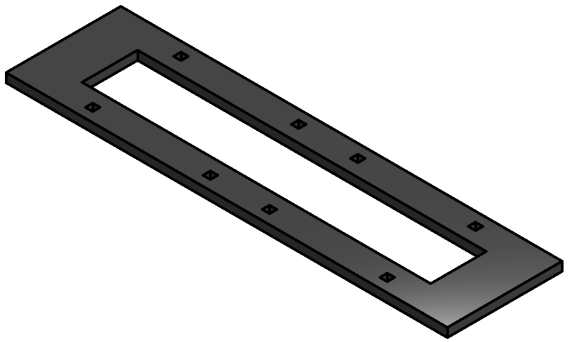
		NAME	SIGNATURE	DATE
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS	DRAWN	SOPHIE CAPLAN		2024-04-15
ANGULAR = ± °	CHECKED			
SURFACE FINISH	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL: PLA		FINISH:	

TITLE: Pentagonal Battery Shaft

SIZE	DWG NO.	REV.
A4		A

SCALE	WEIGHT	SHEET
1:3		1 of 1



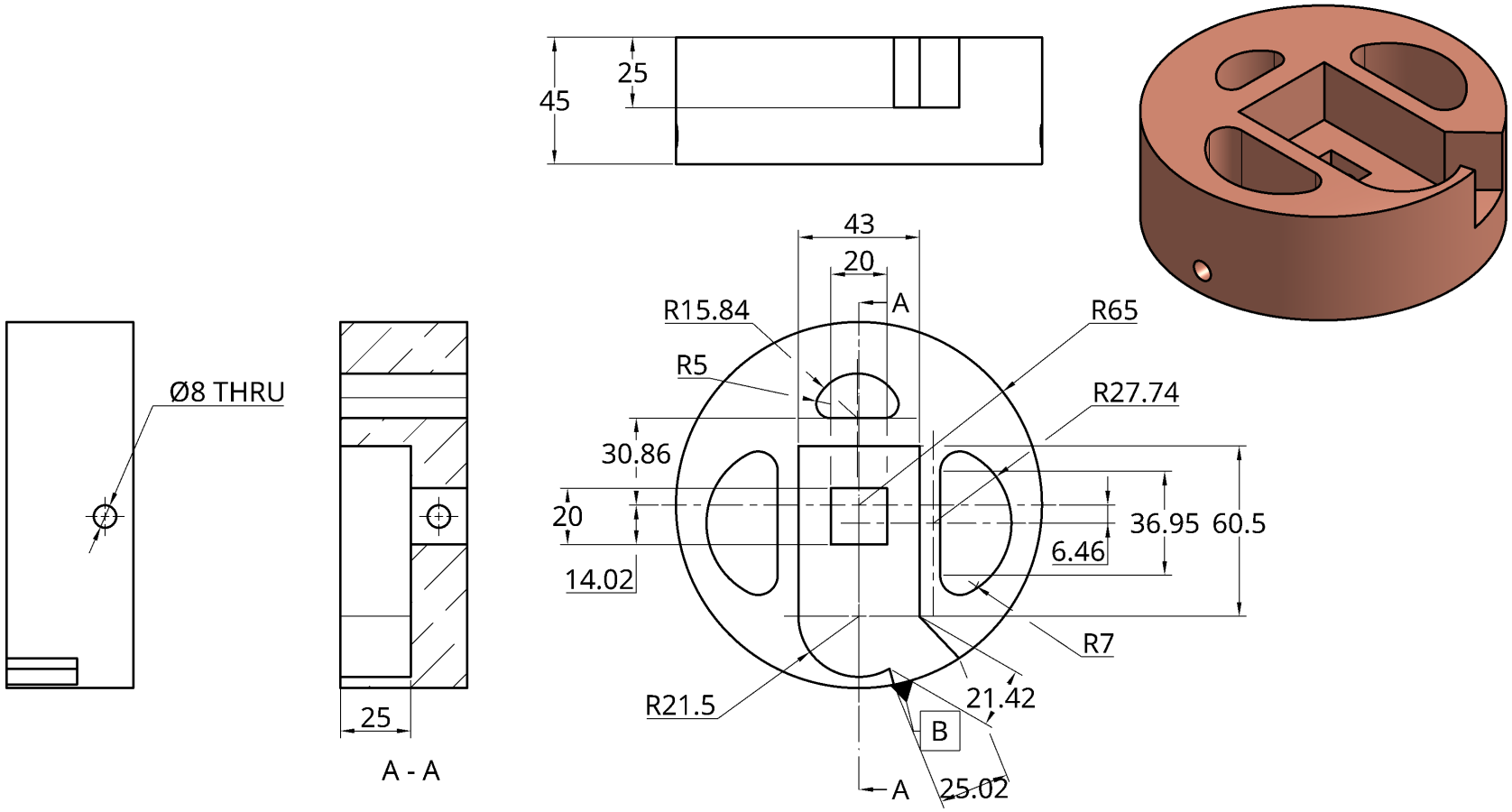


UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS		NAME	SIGNATURE	DATE
ANGULAR = ± °	DRAWN	SOPHIE CAPLAN		2024-04-15
SURFACE FINISH	CHECKED			
	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL	3ml Acrylic	FINISH	

TITLE: Battery Plate

SIZE	DWG NO.	REV.
A4		A

SCALE	WEIGHT	SHEET
1:1.5		1 of 1



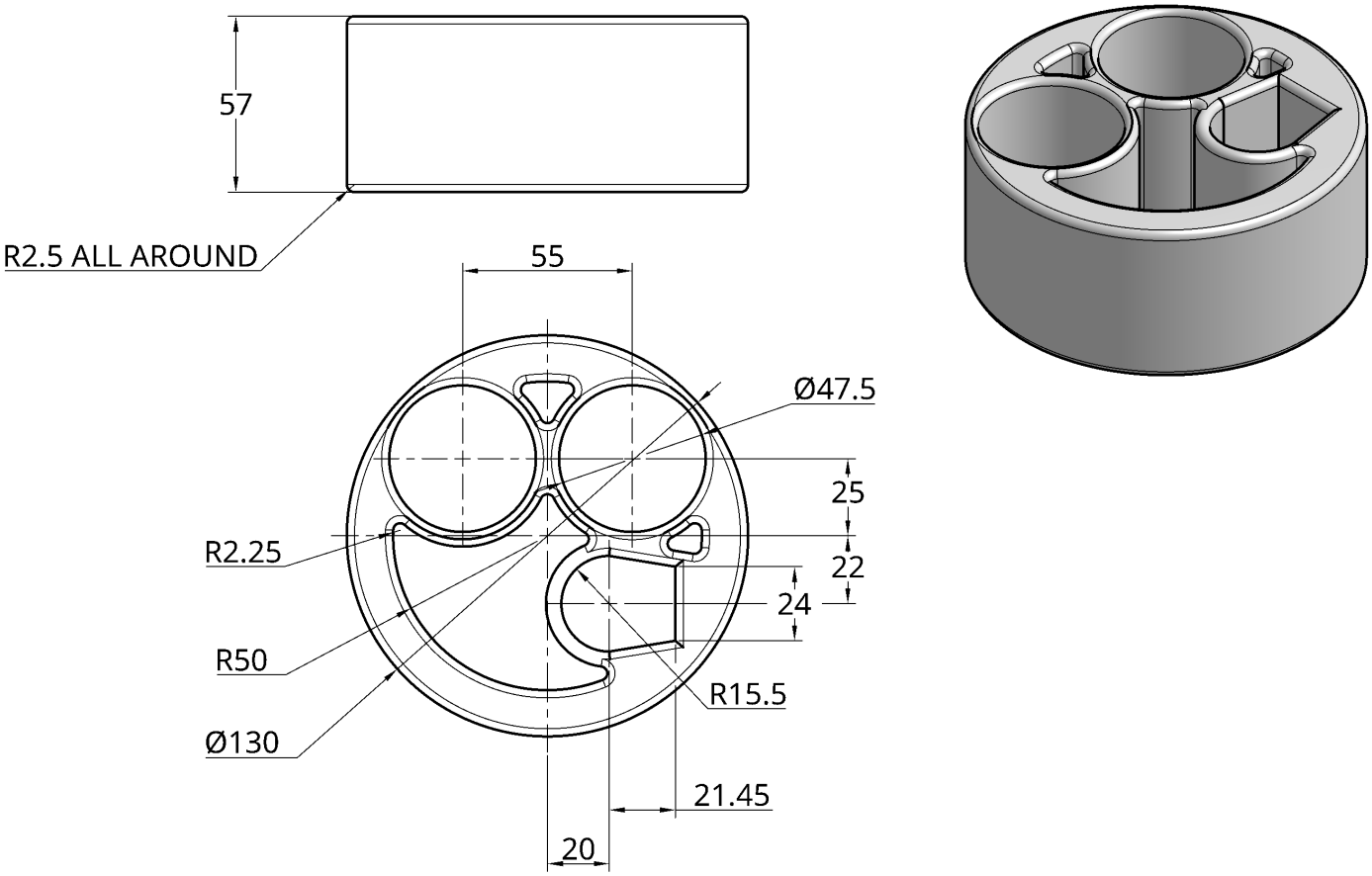
NOTES.
B. CUT-OUT TO ACCOMODATE LINEAR ACTUATOR WIRE

		NAME	SIGNATURE	DATE
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS				
ANGULAR = ± °	DRAWN	SOPHIE CAPLAN		2024-04-15
SURFACE FINISH	CHECKED			
	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL: PLA		FINISH	

TITLE: Base for Battery Linear Actuator

SIZE	DWG NO.	REV.
A4		

SCALE	WEIGHT	SHEET
1:2		1 of 1

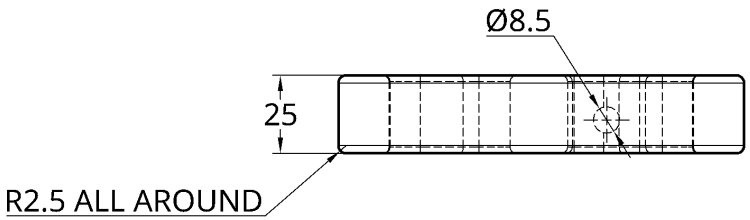


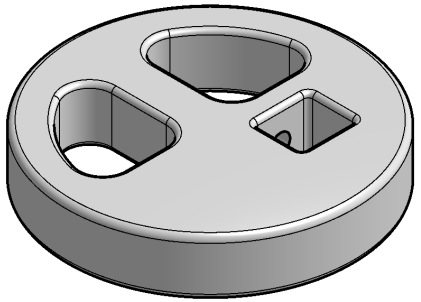
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS		NAME	SIGNATURE	DATE
ANGULAR = ± °	DRAWN	SOPHIE CAPLAN		2024-04-15
SURFACE FINISH √	CHECKED			
	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL: PLA		FINISH:	

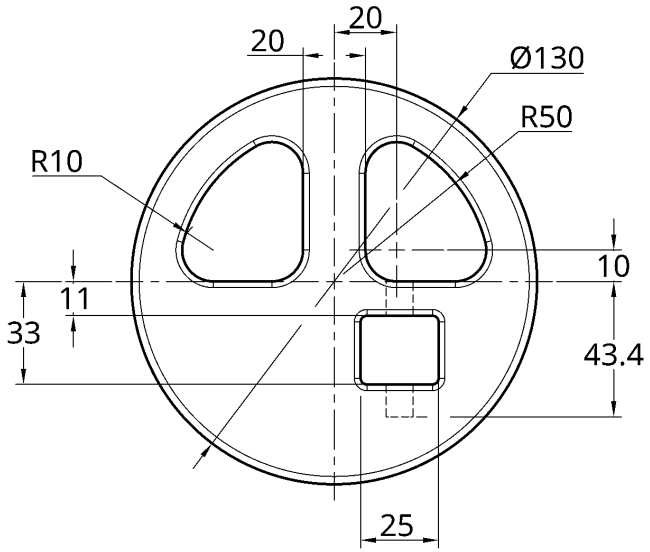
TITLE: Holder for Syringes

SIZE	DWG NO.	REV.
A4		A

SCALE	WEIGHT	SHEET
1:2		1 of 1





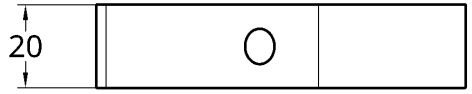


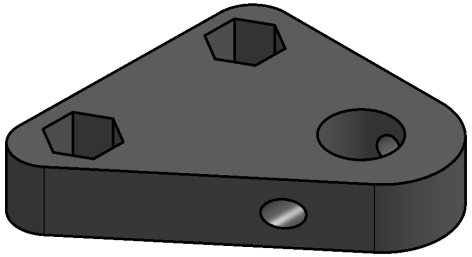
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS		NAME	SIGNATURE	DATE
ANGULAR = ± °	DRAWN	SOPHIE CAPLAN		2024-04-15
SURFACE FINISH	CHECKED			
	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL	PLA	FINISH	

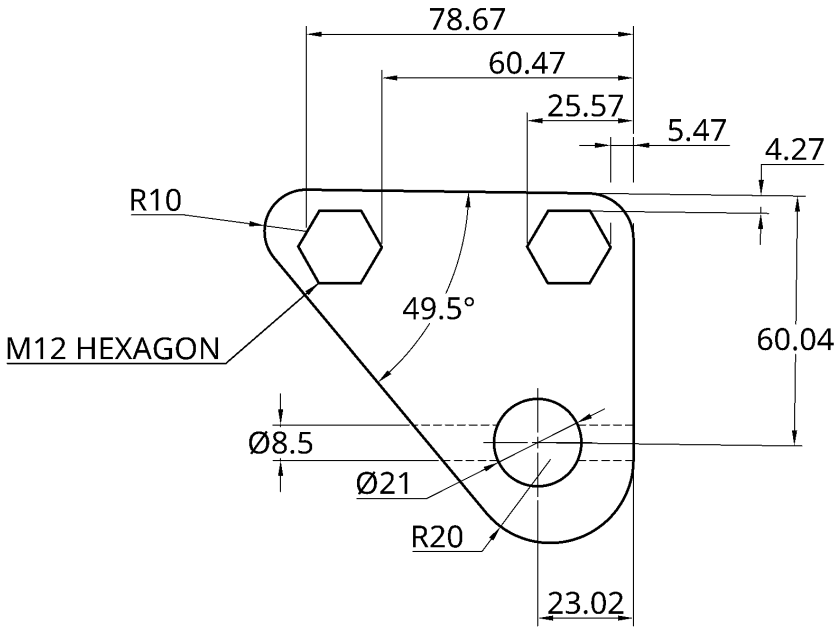
TITLE: Base for Syringe Linear Actuator

SIZE	DWG NO.	REV.
A4		A

SCALE	WEIGHT	SHEET
1:2		1 of 1





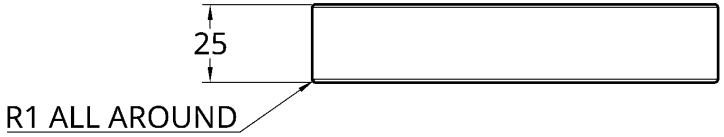


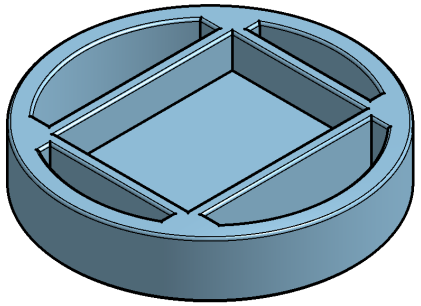
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS ANGULAR = ± ° SURFACE FINISH		NAME	SIGNATURE	DATE
	DRAWN	SOPHIE CAPLAN		2024-04-15
	CHECKED			
	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL: PLA		FINISH	

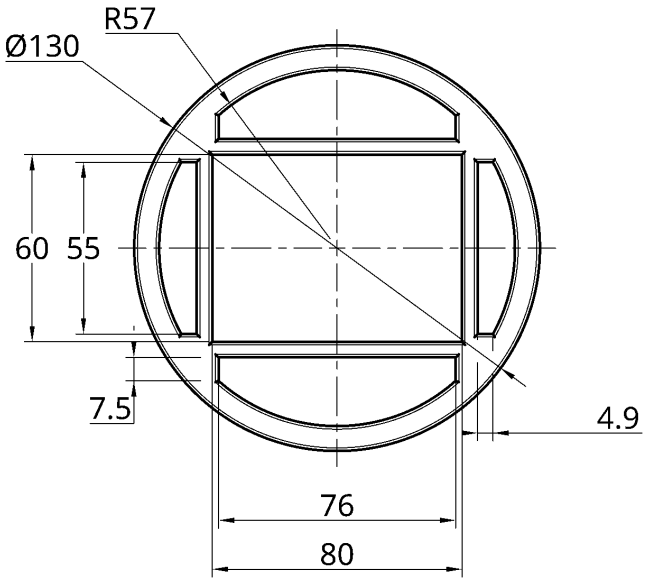
TITLE: Triangle Connector for Syringes

SIZE	DWG NO.	REV.
A4		A

SCALE	WEIGHT	SHEET
1:1.5		1 of 1





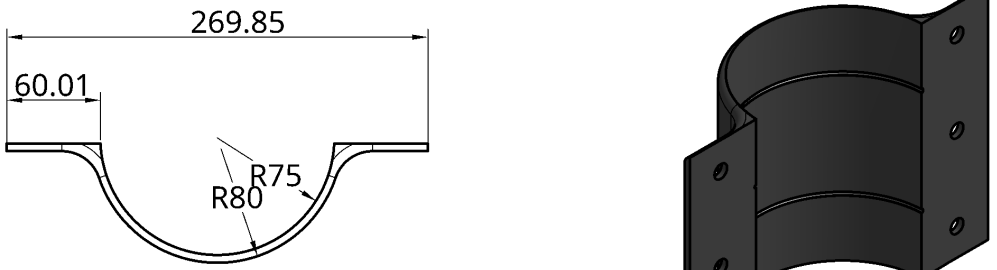


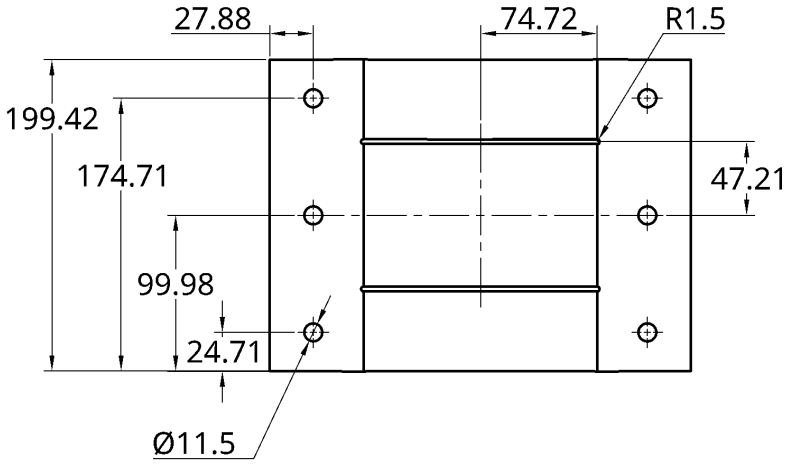
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS		NAME	SIGNATURE	DATE
ANGULAR = ± °	DRAWN	SOPHIE CAPLAN		2024-04-15
SURFACE FINISH	CHECKED			
	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL	PLA	FINISH	

TITLE: Electronics Hub

SIZE	DWG NO.	REV.
A4		A

SCALE	WEIGHT	SHEET
1:2		1 of 1



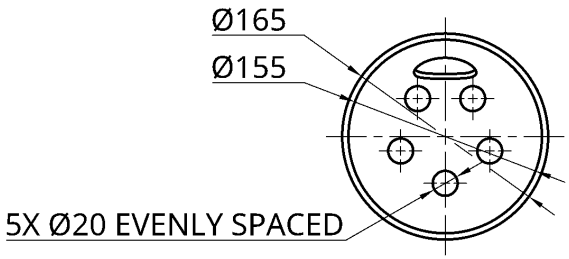


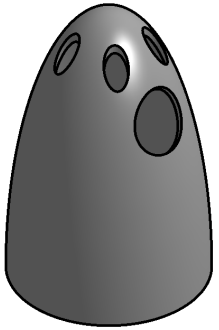
		NAME	SIGNATURE	DATE
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS	DRAWN	SOPHIE CAPLAN		2024-04-15
ANGULAR = ± °	CHECKED			
SURFACE FINISH	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL: PLA		FINISH:	

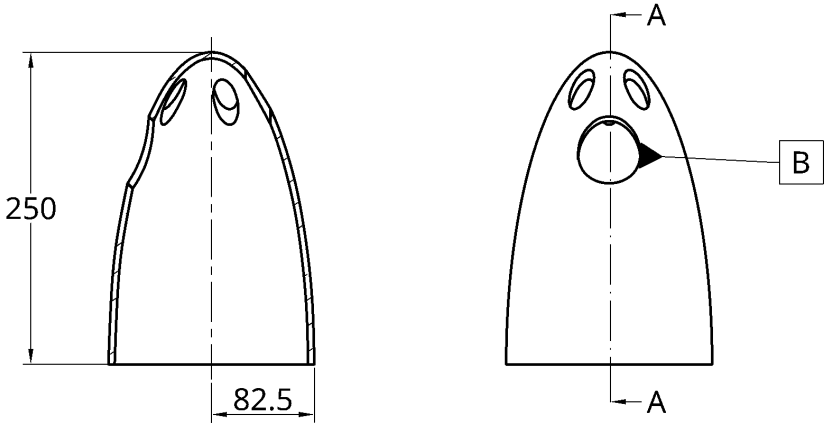
TITLE: Wing Clip

SIZE: A4 | DWG NO. | REV.: A

SCALE: 1:4 | WEIGHT | SHEET: 1 of 1







A - A

NOTES
B. Ø50 HOLE FOR SONAR AT
45° ANGLE TO NOSE PLANE

UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN MILLIMETERS		NAME	SIGNATURE	DATE
ANGULAR = ± °	DRAWN	SOPHIE CAPLAN		2024-04-15
SURFACE FINISH	CHECKED			
	APPROVED			
DO NOT SCALE DRAWING				
BREAK ALL SHARP EDGES AND REMOVE BURRS				
FIRST ANGLE PROJECTION	MATERIAL: PLA		FINISH:	

TITLE: Nose

SIZE	DWG NO.	REV
A4		B

SCALE	WEIGHT	SHEET
1:5		1 of 1

2.5 Bill of Materials

Table 5: Bill of Materials

Item	Name	Qty	Material	Mfg	Vendor	Part Number	Unit Cost	Cost
1	Hull	1	Acrylic	-	The Clear Plastic Shop	XT	£220.10	£220.10
2	End Cap	2	-	-	Toolstation	78470	£14.68	£14.68
3	Battery Module	1	-	-	-	-	-	-
3.1	Mesh	1	PLA	3D Printing	-	-	-	-
3.2	Pentagonal Battery Pack	1	PLA	3D Printing	-	-	-	-
3.3	Pentagonal Battery Pack Clip	1	PLA	3D Printing	-	-	-	-
3.4	Hex Nut Grade A & B M8x1.25	2	Stainless Steel	-	RS	530-769	£0.19	£0.37
3.5	Battery Plates	5	Acrylic	Laser Cutting	-	-	-	-
3.7	Linear Actuator	1	-	-	RS	177-4510	£344.98	£344.98
3.8	Battery Actuator Holder	1	PLA	3D Printing	-	-	-	-
3.9	NiMH D Batteries	10	-	-	Rapid	18-2512	£12.36	£123.60
3.10	Parallel Pin 8 x 80	1	Stainless Steel	-	RS	270-552	£0.24	£0.24
3.11	Plug	3	PLA	3D Printing	-	-	-	-
3.12	Chamfered Plain Washer Normal M2	3	Stainless Steel	-	RS	527-347	£0.02	£0.06
3.13	PH Countersunk Flat Head Screw M2x0.4 x 3	3	Stainless Steel	-	RS	914-1753	£0.07	£0.21
4	Mesh	1	PLA	3D Printing	-	-	-	-
5	Arduino Uno	1	-	-	RS	266-2937	£25.74	£25.74
6	Electronics Hub	1	PLA	3D Printing	-	-	-	-
7	Buoyancy Module	1	-	-	-	-	-	-
7.1	Mesh	4	PLA	3D Printing	-	-	-	-
7.3	Linear Actuator	1	-	-	RS	177-4510	£344.98	£344.98
7.4	Syringe	2	-	-	Rapid	89-8733	£25.09	£50.18
7.6	Syringe Base Plate	1	PLA	3D Printing	-	-	-	-
7.7	Syringe Holder	1	PLA	3D Printing	-	-	-	-
7.8	Triangle Clip	1	PLA	3D Printing	-	-	-	-
7.9	Hex Nut Grade A & B M12x1.75	2	Stainless Steel	-	-	-	-	-
7.10	Tubing	2	PVC	-	RS	914.5519	£13.22	£13.22
7.11	Plug	3	PLA	3D Printing	-	-	-	-
7.12	PH Countersunk Flat Head Screw M2x0.4 x 3	3	Stainless Steel	-	RS	914-1753	£0.07	£0.21
7.13	Chamfered Plain Washer Normal M2	3	Stainless Steel	-	RS	248-4595	£0.04	£0.13
8	Nose	1	PLA	3D Printing	-	-	-	-
9	Sonar	1	-	-	Robotshop	RB-Blu-259	£390.01	£390.01
10	Wing Clip	2	PLA	3D Printing	-	-	-	-
11	Wings	2	Aluminium	-	RS	778-1661	£193.45	£386.90
12	O-Rings	2	Nitrile	-	RS	255-6835	£2.46	£4.92
13	PH Countersunk Flat Head Screw M10x1.5 x 45	6	Stainless Steel	-	RS	508-0994	£1.31	£7.85
14	Chamfered Plain Washer Normal M10	12	Stainless Steel	-	RS	527-640	£0.06	£0.72
14	Chamfered Plain Washer Normal M10	12	Stainless Steel	-	RS	527-640	£0.06	£0.72
15	Hex Nut Grade A & B M10x1.5	6	Stainless Steel	-	RS	530-775	£0.32	£1.90

2.6 Description of Computer Code

2.6.1 Control System Overview

Fig.9. shows a visual representation of the glider's operating loop. It has two primary states - diving and surfacing. The glider moves from the diving state to its surfacing state when it meets certain conditions. These are:

- It locates the lakebed
- It reaches its maximum depth

The buoyancy engine and pitch control mechanisms are controlled based on the glider's state. Details of the code used to control the glider are given in section 2.6.2. Table 6 also provides the details of the pin layout.

2.6.2 Electronic System Overview

The autonomous underwater glider is controlled by an Arduino Uno R4, interfaced with a BMX055 IMU (Inertial Measurement Unit) for orientation and motion sensing, a Ping Sonar for distance measurement and obstacle detection underwater, a TSYS01 temperature sensor for measuring temperature gradients and BTS7960B motor drivers to operate two linear actuators. These actuators are pivotal for buoyancy control via syringe mechanisms and pitch control by adjusting the position of a battery module. The system is modular, enhancing maintainability and scalability.

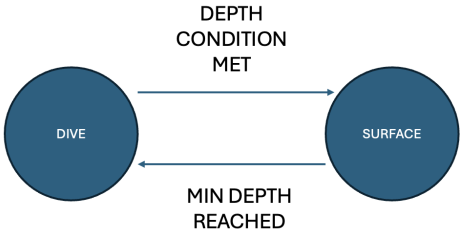


Figure 9: Control Scheme

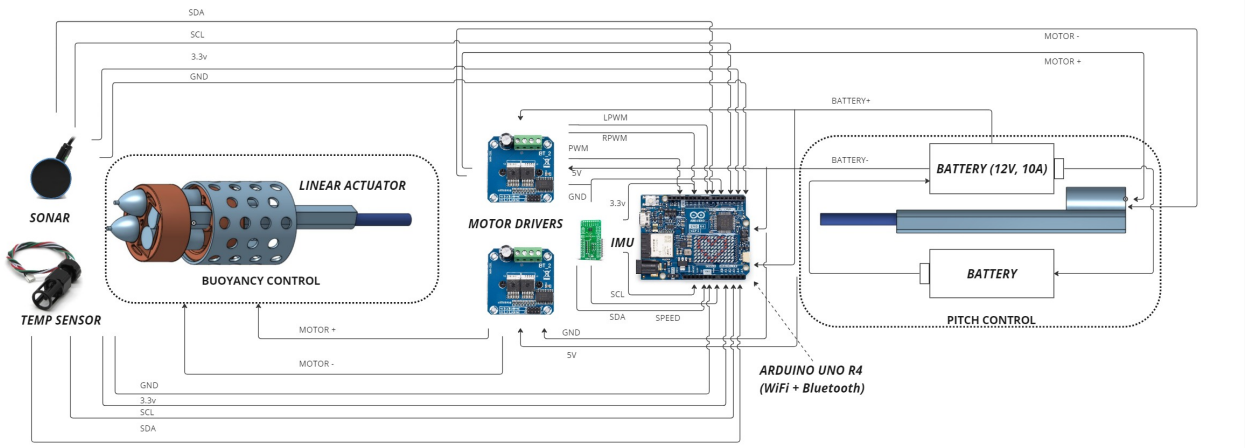


Figure 10: Wiring Overview

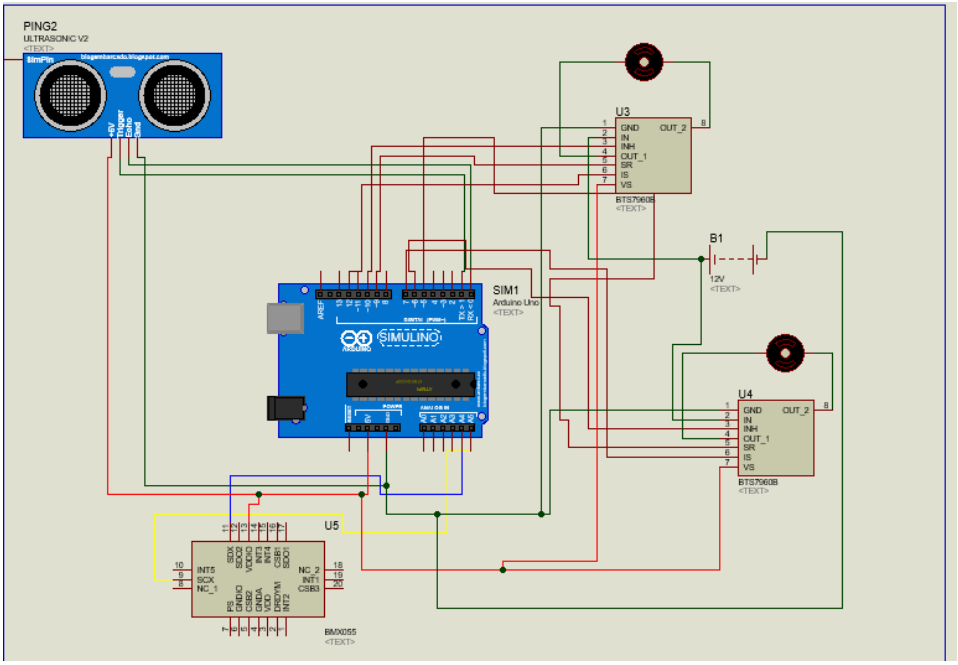


Figure 11: Circuit Layout

Table 6: Pin Layout

Arduino Pin	Buoyancy engine	Battery Module	SONAR Pin	IMU
5v	enable	enable		enable
3.3 v			Power	
GND			GND	GND
0			TX	
1			RX	
2				
3		PWM		
4				
5		LPWM		
6		RPWM		
7				
8				
9	RPWM			
10	LPWM			
11				
12	PWM			
13				
A0				SCL
A1				SDA

The system is structured in a modular fashion with dedicated header files for the IMU and Sonar, facilitating independent testing and scalability of these components. The overall system architecture supports easy integration of new sensors, enhancing the glider's functionality and adaptability.

An overview of the wiring can be seen in fig 10 along with the circuit diagram in fig.11. In the circuit diagram, the brown wires show connections between the Arduino and the motor drivers, the red wires show connections between the Arduino 5v and components that require power. The yellow wire is the SCL wire for the IMU and the blue wire is the SDA wire for the IMU.

2.6.3 Software Architecture

- Inertial Measurement Module

  The IMU program is primarily focused on integrating the BMX055 IMU sensor data with the Autonomous Underwater Glider. The BMX055, a compact 9-axis IMU, provides three-dimensional measurements across its accelerometer, gyroscope, and magnetometer. The accelerometer measures linear acceleration along the x, y, and z axes, useful for detecting changes in velocity and orientation due to external forces. The gyroscope captures the rotational motion around the three axes, crucial for determining orientation changes and assisting in dynamic positioning and navigation corrections and the magnetometer offers heading information based on Earth's magnetic fields, which is vital for directional control and ensures the glider maintains its bearing. Due to the absence of BMX055 header files compatible with the Arduino Uno R4, the code presented below demonstrates a complete implementation.

  This code defines a class `IMU` that interfaces with an IMU sensor on an Arduino platform. The header file declares the class and its methods, while the source file implements these methods.

  The header file `IMU.h` defines a class `IMU` which is used to interface with an IMU sensor. Below is the detailed breakdown of its definition:

```
#ifndef IMU_h
#define IMU_h

#include "Arduino.h"
#include "Wire.h"
```

```
6
7  class IMU {
8  public:
9      IMU(); // Constructor
10     void begin(); // Initialize IMU settings
11     void readAccelerometer(int &ax, int &ay, int &az);
12     void readGyroscope(int &gx, int &gy, int &gz);
13     void readMagnetometer(int &mx, int &my, int &mz);
14
15 private:
16     void writeRegister(byte addr, byte reg, byte data);
17     void readData(byte addr, byte reg, byte num, byte *buf);
18 };
19
20 #endif
```

Explanation

- Preprocessor Directives: Prevents multiple inclusions of the header file.
- Includes: The code includes headers for Arduino and I2C communication (`"Arduino.h"` and `"Wire.h"`).
- IMU Class Definition: The `IMU` class has both public and private members:
  - Constructor: Defines an initialiser for the IMU object.
  - Sensor Methods: Methods to begin IMU operation, read accelerometer, gyroscope, and magnetometer data.
  - Private Methods: Methods to write and read data to/from the IMU device registers.

The `IMU.cpp` file provides the implementations for the methods declared in the `IMU.h` header file. Below is the detailed implementation:

```
1  #include "IMU.h"
2
3  IMU::IMU() {}
4
5  void IMU::begin() {
6      Wire.begin(); // Start I2C
7      Serial.begin(9600); // Start serial for output
8
9      // Initialize Accelerometer
10     writeRegister(0x18, 0x0F, 0x03); // Set range
11     writeRegister(0x18, 0x10, 0x08); // Set bandwidth
12
13     // Initialize Gyroscope
14     writeRegister(0x68, 0x0F, 0x04); // Set range
15     writeRegister(0x68, 0x10, 0x07); // Set bandwidth
16
17     // Initialize Magnetometer
18     writeRegister(0x10, 0x4B, 0x83); // Soft reset
19     delay(50); // Wait for reset to complete
20     writeRegister(0x10, 0x4C, 0x00); // Normal mode
21 }
22
23 void IMU::writeRegister(byte addr, byte reg, byte data) {
24     Wire.beginTransmission(addr);
25     Wire.write(reg);
26     Wire.write(data);
27     Wire.endTransmission();
28 }
29
30 void IMU::readData(byte addr, byte reg, byte num, byte *buf) {
31     Wire.beginTransmission(addr);
32     Wire.write(reg);
```

```
    Wire.endTransmission();
    Wire.requestFrom(addr, num);
    int i = 0;
    while (Wire.available()) {
        buf[i++] = Wire.read();
    }
}

void IMU::readAccelerometer(int &ax, int &ay, int &az) {
    byte buf[6];
    readData(0x18, 0x02, 6, buf);
    ax = (buf[1] << 8) | buf[0];
    ay = (buf[3] << 8) | buf[2];
    az = (buf[5] << 8) | buf[4];
}

void IMU::readGyroscope(int &gx, int &gy, int &gz) {
    byte buf[6];
    readData(0x68, 0x02, 6, buf);
    gx = (buf[1] << 8) | buf[0];
    gy = (buf[3] << 8) | buf[2];
    gz = (buf[5] << 8) | buf[4];
}

void IMU::readMagnetometer(int &mx, int &my, int &mz) {
    byte buf[6];
    readData(0x10, 0x42, 6, buf);
    mx = (buf[1] << 8) | buf[0];
    my = (buf[3] << 8) | buf[2];
    mz = (buf[5] << 8) | buf[4];
}
```

Explanation

- Constructor and Initialisation: The constructor is empty. The `begin` method sets up the I2C communication, initialises serial output, and configures the accelerometer, gyroscope, and magnetometer with specific register settings.
- Data Access Methods: The `writeRegister` and `readData` methods facilitate low-level communication with the IMU sensor. `readAccelerometer`, `readGyroscope`, and `readMagnetometer` methods extract and format the sensor data.

The `begin()` function initialises the I2C communication, sets up serial communication for debugging, and configures the accelerometer, gyroscope, and magnetometer with specific settings for range and bandwidth.

- Sonar Module

  This module manages the operations of the Ping sonar sensor, which is crucial for distance measurement. This sonar system helps the glider detect obstacles and measure the distance to the lakebed or obstacles in its path. The script initialises the sonar sensor and configures it to send out pulses and calculates the distance by measuring the time it takes for the echo to return.

  The header file `Sonar.h` defines a class `Sonar` which is used to interface with a sonar sensor through a hardware serial port. Below is the detailed breakdown of its definition:

```
#ifndef Sonar_h
#define Sonar_h

#include "Arduino.h"
#include "ping1d.h"
#include "HardwareSerial.h"

```

```
class Sonar {
public:
    Sonar(HardwareSerial& serialPort); // Constructor that takes a reference to
     a Serial port
    bool initialize();
    int getDistance();

private:
    Ping1D ping;
};

#endif
```

Explanation

- Preprocessor Directives: The code starts with preprocessor directives to prevent multiple inclusion of the header file (`#ifndef`, `#define`, `#endif`).
- Includes: It includes necessary headers from the Arduino framework and other libraries needed for the sonar sensor (`#include "Arduino.h"`, `#include "ping1d.h"`, `#include "HardwareSerial.h"`).
- Sonar Class Definition: The `Sonar` class is defined with public and private members:
  - Constructor: Accepts a reference to a `HardwareSerial` object, which facilitates communication with the sonar sensor.
  - initialize(): A method returning a boolean indicating successful initialization of the sensor.
  - getDistance(): A method to retrieve the distance measurement from the sensor.
  - Private Member: A `Ping1D` object named `ping` to manage sensor data.

The `Sonar.cpp` file provides the implementations of the methods declared in the `Sonar.h` header file. Below is the detailed implementation:

```
#include "Sonar.h"

Sonar::Sonar(HardwareSerial& serialPort) : ping(serialPort) {}

bool Sonar::initialize() {
    return ping.initialize();
}

int Sonar::getDistance() {
    if (ping.update()) {
        return ping.getDistance();
    }
    return -1; // Return -1 if no new data is available
}

bool Sonar::obstacleAvoid()
    thresholdDistance = 1500;
    if (ping.getDistance()<=thresholdDistance)
    {
        obsDist = ping.getDistance();
        startTime = millis();
        if (ping.getDistance() < obsDist && (currentTime -startTime) >= 3000)
        {
            return 1
        }
    }
```

Explanation

- Constructor Implementation: The constructor initialises the `Ping1D` object `ping` with the provided `HardwareSerial` reference, preparing it for communication with the sonar sensor.
- initialise Method: This method simply forwards the call to `ping`'s `initialize` method, which configures the sensor settings.
- getDistance Method: The `getDistance` method first calls `ping.update()` to refresh the sensor data. If new data is available, it returns the distance measured by the sensor. Otherwise, it returns `-1` indicating no new data is available.

The Sonar was primarily used for obstacle detection and avoidance. Since lakes have variable depth, it is important to be able to detect the approaching lakebed and take evasive manoeuvres. To do so the obstacleAvoid() function was used. It triggers a timer if the distance measured by the sonar is less than the threshold distance of 1.5m. If the distance measured after the time set is less than the threshold distance, the glider switches from the diving state to the surfacing state.

- Temperature Module

  The code used is based on the one provided by Blue Robotics for their TSYS01 temperature sensor. It uses the Wire library to communicate with the temperature sensor using the I2C protocol. The script reads the analogue data from the sensor and then calculates the temperature at a fixed rate.

  The header file `"TSYS01.h"` defines a class `"TSYS01"` which handles communications with the temperature sensor over I2C protocol.

```
#ifndef TSYS01_H_BLUEROBOTICS
#define TSYS01_H_BLUEROBOTICS

#include "Arduino.h"

class TSYS01 {
public:

  TSYS01();

  bool init();

  void read();

  /** This function loads the datasheet test case values to verify that
   *  calculations are working correctly. No example checksum is provided
   *  so the checksum test may fail.
   */
  void readTestCase();

  /** Temperature returned in deg C.
   */
  float temperature();

private:
  uint16_t C[8];
  uint32_t D1;
  float TEMP;
  uint32_t adc;

  /** Performs calculations per the sensor data sheet for conversion and
   *  second order compensation.
   */
  void calculate();

};
```

```
37
38 #endif
```

Explanation:

- Preprocessor Directives: The code starts with preprocessor directives to prevent multiple inclusion of the header file (`#ifndef`, `#define`, `#endif`).
- Include: The Arduino framework is included in this header file (`#include \Arduino.h"`)
- TSYS01 Class Definition: The `TSYS01` class is defined with public methods for the initialisation of the sensor and reading of values, as well as the `TSYS01()` constructor. The class also has private members and a private method `calculate()` to calculate the temperature based on the analogue readings.

The `TSYS01.cpp` file provides the implementations of the methods declared in the `TSYS01.h` header file. Below is the detailed implementation:

```
1  #include "TSYS01.h"
2  #include <Wire.h>
3
4  #define TSYS01_ADDR                          0x77
5  #define TSYS01_RESET                         0x1E
6  #define TSYS01_ADC_READ                      0x00
7  #define TSYS01_ADC_TEMP_CONV                 0x48
8  #define TSYS01_PROM_READ                     0XA0
9
10 TSYS01::TSYS01() {
11
12 }
13
14 bool TSYS01::init() {
15   // Reset the TSYS01, per datasheet
16   Wire1.beginTransmission(TSYS01_ADDR);
17   Wire1.write(TSYS01_RESET);
18   Wire1.endTransmission();
19
20   delay(10);
21   int received_bytes = 0;
22     // Read calibration values
23   for ( uint8_t i = 0 ; i < 8 ; i++ ) {
24     Wire1.beginTransmission(TSYS01_ADDR);
25     Wire1.write(TSYS01_PROM_READ+i*2);
26     Wire1.endTransmission();
27
28     received_bytes += Wire1.requestFrom(TSYS01_ADDR,2);
29     C[i] = (Wire1.read() << 8) | Wire1.read();
30   }
31   return received_bytes > 0;
32
33 }
34
35 void TSYS01::read() {
36
37   Wire1.beginTransmission(TSYS01_ADDR);
38   Wire1.write(TSYS01_ADC_TEMP_CONV);
39   Wire1.endTransmission();
40
41   delay(10); // Max conversion time per datasheet
42
43   Wire1.beginTransmission(TSYS01_ADDR);
44   Wire1.write(TSYS01_ADC_READ);
45   Wire1.endTransmission();
46
47   Wire1.requestFrom(TSYS01_ADDR,3);
```

```
  D1 = 0;
  D1 = Wire1.read();
  D1 = (D1 << 8) | Wire1.read();
  D1 = (D1 << 8) | Wire1.read();

  calculate();
}

void TSYS01::readTestCase() {
  C[0] = 0;
  C[1] = 28446;  //0xA2 K4
  C[2] = 24926;  //0XA4 k3
  C[3] = 36016;  //0XA6 K2
  C[4] = 32791;  //0XA8 K1
  C[5] = 40781;  //0XAA K0
  C[6] = 0;
  C[7] = 0;

  D1 = 9378708.0f;

  adc = D1/256;

  calculate();
}

void TSYS01::calculate() {
  adc = D1/256;

TEMP = (-2) * float(C[1]) / 1000000000000000000.0f * pow(adc,4) +
        4 * float(C[2]) / 10000000000000000.0f * pow(adc,3) +
    (-2) * float(C[3]) / 100000000000.0f * pow(adc,2) +
        1 * float(C[4]) / 1000000.0f * adc +
      (-1.5) * float(C[5]) / 100 ;

}

float TSYS01::temperature() {
  return TEMP;
}
```

Explanation of the TSYS01 methods:

- `TSYS01()`: The constructor is empty and serves only to create a TSYS01 object.
- `init()`: The initialisation method starts communication with the sensor over I2C protocol using the Wire library and proceeds to calibrate the sensor.
- `calculate()`: This method calculated the read temperature in degrees Celsius from the analogue readings.
- `read()`: The `read` method receives the sensor readings through I2C and then calls the `calculate` method to calculate the temperature in degrees Celsius.
- `temperature()`: The temperature method returns the value of `TEMP` set by `calculate()`.
- `readTestCase()`: This method is only used to test `calculate()`.

- Communication Module

  The Arduino Uno R4 Wi-Fi allows for Bluetooth Low Energy (BLE) communication with other BLE capable devices. Using BLE allows for fast and reliable communications without using much energy. The following presents a breakdown of the BLE functionalities.

```
// Bluetooth Definitions
#define SERVICE_UUID "180A"
#define CONTROL_CHARACTERISTIC_UUID "2A57"
BLEService gliderService(SERVICE_UUID);
```

```
BLEByteCharacteristic ControlCharacteristic(CONTROL_CHARACTERISTIC_UUID,
    BLERead | BLEWrite);
```

The script creates a BLE service with a Universally Unique Identifier (UUID) so that it can be differentiated from other BLE devices. It then characterises the service as a Read/Write connection, meaning that it can send and receive data from any paired device.

```
// Initialize Bluetooth LE
if (!BLE.begin()) {
  Serial.println("Starting BLE failed!");
  while (1);
}
BLE.setLocalName("Glider");
BLE.setAdvertisedService(gliderService);
gliderService.addCharacteristic(ControlCharacteristic);
BLE.addService(gliderService);
ControlCharacteristic.writeValue(0);
BLE.advertise();
```

During setup, the Bluetooth service is initialised, and its name is set to "Glider" for easy identification. The communicated value is initialised as 0 and the characteristics are added to the service. The service is thereby advertised using the UUID. At this point, the script listens for connections from BLE devices. Once a connection is detected, communications begin between the paired devices (glider and controller).

```
if (BLE.connected()) {
  if (ControlCharacteristic.written()) {
    manualControl = ControlCharacteristic.value() != 0; // Check for non-zero
    value for manual control
  }
}

if (!manualControl) {
    // Autonomous navigation
} else {
  byte command = ControlCharacteristic.value();
  switch (command) {
    // User defined commands
  }
}
```

While the controller is connected, the script checks whether the glider has been instructed to pass into manual control. If so, the glider will follow user set instructions defined in a switch case, each instruction corresponding to a case. Otherwise, the glider is in autonomous navigation and relies on sonar and IMU readings for buoyancy control.

- Motor Control

  The BTS7960 is a robust high-power motor driver, capable of handling currents up to 43A and is equipped with heatsink. It is used to operate the linear actuators that control buoyancy and pitch.The motor driver is powered using 5v and the RPWM (Right PWM) – Connects to an Arduino PWM pin for driving the motor in one direction. LPWM (Left PWM) – Connects to another PWM pin for driving in the opposite direction. R_EN and L_EN (Enable pins) – These are connected to digital outputs on Arduino for enabling or disabling the driver channels. Through these connections, the Arduino can precisely control the direction and speed of the actuators, adjusting the glider's buoyancy by modifying the volume of water in the syringes and altering its pitch through the movement of the battery module.

- Timing and Control Logic

  The final Arduino script is the culmination of integrating sensor inputs and actuator outputs using a sophisticated control loop. Timing in this context is crucial as it affects the responsiveness and stability of the control system.

This Arduino sketch is designed for a dynamic PID control system linked with Bluetooth, specifically for controlling an underwater vehicle's buoyancy and pitch via actuators. Detailed explanation of the sketch is as follows:

Header Inclusion and Pin Definition

```
#include "IMU.h"
#include "Sonar.h"
#include "Arduino.h"
#include <PID_v1.h>
#include <ArduinoBLE.h>

// Pin definitions for actuators
#define RPWM1 9   // Right PWM for buoyancy control
#define LPWM1 10  // Left PWM for buoyancy control
#define PWM1 12   // PWM for buoyancy control
#define RPWM2 6   // Right PWM for pitch control
#define LPWM2 5   // Left PWM for pitch control
#define PWM2 3    // PWM for pitch control
```

Bluetooth Service and Characteristic Definition

```
// Bluetooth Definitions
#define SERVICE_UUID "180A"
#define CONTROL_CHARACTERISTIC_UUID "2A57"
BLEService gliderService(SERVICE_UUID);
BLEByteCharacteristic ControlCharacteristic(CONTROL_CHARACTERISTIC_UUID,
    BLERead | BLEWrite);
```

Initialization of Sensors and Actuators

```
IMU imu;
Sonar sonar(Serial1);
```

Control Constants and PID Configuration

```
const float proximityThreshold = 1500.0; // 1.5 meters in mm
const unsigned long maxBuoyancyDuration = 9000; // 9 seconds
const unsigned long maxPitchDuration = 18000; // 18 seconds
unsigned long buoyancyStartTime, pitchStartTime;

double setpointBuoyancy, inputBuoyancy, outputBuoyancy;
double setpointPitch, inputPitch, outputPitch;
double KpBuoyancy = 2.0, KiBuoyancy = 0.5, KdBuoyancy = 1.0;
double KpPitch = 1.5, KiPitch = 0.4, KdPitch = 0.9;

PID buoyancyPID(&inputBuoyancy, &outputBuoyancy, &setpointBuoyancy, KpBuoyancy,
     KiBuoyancy, KdBuoyancy, DIRECT);
PID pitchPID(&inputPitch, &outputPitch, &setpointPitch, KpPitch, KiPitch,
    KdPitch, DIRECT);

bool manualControl = false;
```

Setup Function

```
void setup() {
  Serial.begin(9600);
  imu.begin();
  // Initialize sonar with retry logic
  int initAttempts = 0;
  while (!sonar.initialize() && initAttempts < 3) {
    Serial.println("Sonar failed to initialize!");
    delay(2000);
    initAttempts++;
  }
  pinMode(RPWM1, OUTPUT);
  pinMode(LPWM1, OUTPUT);
```

```
  pinMode(PWM1, OUTPUT);
  pinMode(RPWM2, OUTPUT);
  pinMode(LPWM2, OUTPUT);
  pinMode(PWM2, OUTPUT);
  // Start Bluetooth service
  if (!BLE.begin()) {
    Serial.println("Starting BLE failed!");
    while (1);
  }
  BLE.setLocalName("Glider");
  BLE.setAdvertisedService(gliderService);
  gliderService.addCharacteristic(ControlCharacteristic);
  BLE.addService(gliderService);
  ControlCharacteristic.writeValue(0);
  BLE.advertise();
  setpointBuoyancy = 100; // Target distance from the bottom in cm
  setpointPitch = 0; // Target pitch level in degrees
  buoyancyPID.SetMode(AUTOMATIC);
  pitchPID.SetMode(AUTOMATIC);
}
```

Main Loop and Control Logic

```
void loop() {
  if (BLE.connected()) {
    if (ControlCharacteristic.written()) {
      manualControl = ControlCharacteristic.value() != 0;
    }
  }
  if (!manualControl) {
    int ax, ay, az, gx, gy, gz, mx, my, mz;
    bool imuAvailable = imu.readAccelerometer(ax, ay, az) && imu.readGyroscope(
   gx, gy, gz) && imu.readMagnetometer(mx, my, mz);
    int distance = sonar.getDistance();
    if (distance > 0 && distance <= proximityThreshold) {
      actuatorRetract(LPWM1, RPWM1, PWM1);
    } else if (!imuAvailable) {
      actuatorRetract(LPWM1, RPWM1, PWM1);
    }
    inputBuoyancy = distance / 10.0; // Convert mm to cm
    inputPitch = ay;
    buoyancyPID.Compute();
    pitchPID.Compute();
    if (outputBuoyancy > 0) {
      actuatorExtend(LPWM1, RPWM1, PWM1);
    } else {
      actuatorRetract(LPWM1, RPWM1, PWM1);
    }
    if (outputPitch > 0) {
      actuatorExtend(LPWM2, RPWM2, PWM2);
    } else {
      actuatorRetract(LPWM2, RPWM2, PWM2);
    }
    // Failsafe timing checks
    if (millis() - buoyancyStartTime > maxBuoyancyDuration) {
      actuatorStop(LPWM1, RPWM1, PWM1);
    }
    if (millis() - pitchStartTime > maxPitchDuration) {
      actuatorStop(LPWM2, RPWM2, PWM2);
    }
  } else {
    byte command = ControlCharacteristic.value();
    switch (command) {
      case 1: actuatorExtend(LPWM1, RPWM1, PWM1); break;
      case 2: actuatorRetract(LPWM1, RPWM1, PWM1); break;
      case 3: actuatorExtend(LPWM2, RPWM2, PWM2); break;
```

```
      case 4: actuatorRetract(LPWM2, RPWM2, PWM2); break;
      case 0: actuatorStop(LPWM1, RPWM1, PWM1);
              actuatorStop(LPWM2, RPWM2, PWM2); break;
    }
  }
}
```

Actuator Control Functions

Detailed functions for actuator control including extending, retracting, and stopping based on PID outputs or manual commands.

```
void actuatorExtend(int LPWM, int RPWM, int PWM) {
  digitalWrite(LPWM, LOW);
  digitalWrite(RPWM, HIGH);
  analogWrite(PWM, 255); // Fully extend
  Serial.println("Actuator is Extending");
  buoyancyStartTime = millis(); // Restart timer when extending
}

void actuatorRetract(int LPWM, int RPWM, int PWM) {
  digitalWrite(LPWM, HIGH);
  digitalWrite(RPWM, LOW);
  analogWrite(PWM, 255); // Fully retract
  Serial.println("Actuator is Retracting");
  buoyancyStartTime = millis(); // Restart timer when retracting
}

void actuatorStop(int LPWM, int RPWM, int PWM) {
  digitalWrite(LPWM, LOW);
  digitalWrite(RPWM, LOW);
  analogWrite(PWM, 0); // Stop
  Serial.println("Actuator Stopped");
}
```

The code provided was used in the control implementation of the glider, utilising a range of onboard sensors and actuators to navigate and maintain positioning within the underwater environment. The software framework integrates with several libraries, including those for handling Inertial Measurement Units (IMU), sonar sensors, PID control algorithms, and Bluetooth Low Energy (BLE) for communication.

The code defines several pin assignments on the Arduino for Pulse Width Modulation (PWM) control of actuators responsible for buoyancy and pitch adjustments. Two unique UUIDs are specified for the BLE service and characteristics, enabling remote command and control over the vehicle (this cannot be used while the AUG is underwater). Constants are defined to set proximity thresholds for obstacle avoidance and maximum operation times for actuators to prevent overuse and potential failure.

Global objects are instantiated for the BLE service, IMU, and sonar sensors, facilitating data acquisition and environmental interaction. The PID controllers for buoyancy and pitch are initialised with specific tuning parameters to optimise response times and stability of the vehicle's movements. Additionally, variables are set to record the timing of operations, providing a basis for timeout functionalities and operational safety protocols.

In the setup() function, initialisation sequences are carried out for serial communications, sensors, and actuators, along with the necessary configurations for BLE functionality. This setup ensures that the vehicle is ready to receive and execute commands upon startup. The PID controllers are set to automatic mode, priming the system for real-time adjustment based on sensor inputs.

In the main operational loop, if the vehicle is not under manual control, it autonomously reads from the IMU and sonar to guide its actions. Depending on the proximity to obstacles and the availability of sensor data, actuators are controlled to adjust buoyancy and pitch. The PID controllers dynamically compute the required outputs to maintain set targets for distance from obstacles and

desired pitch angles. Safety mechanisms are enforced through timeouts, ensuring that actuators do not operate beyond their intended duration.

Actuator control functions—actuatorExtend(), actuatorRetract(), and actuatorStop()—manipulate the defined PWM pins to achieve the necessary mechanical movements. These functions are pivotal for the physical response of the vehicle, extending or retracting actuators based on the computed needs from the PID controllers or direct manual commands. The functionality of these methods is logged via serial outputs to provide real-time feedback and diagnostics.

Overall, the software architecture described provides a robust framework for the operation of an autonomous submersible vehicle, integrating advanced control methodologies and communication technologies to ensure precise and safe underwater navigation and maneuverability.

2.7 Photos of Final Product

Figure 12: Glider

Figure 13: Glider Nose

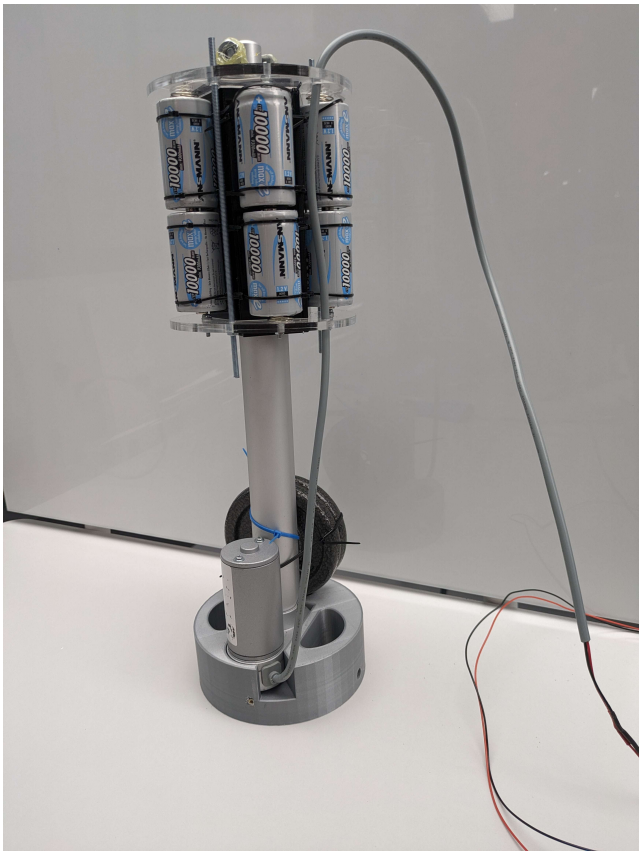


Figure 14: Battery Module

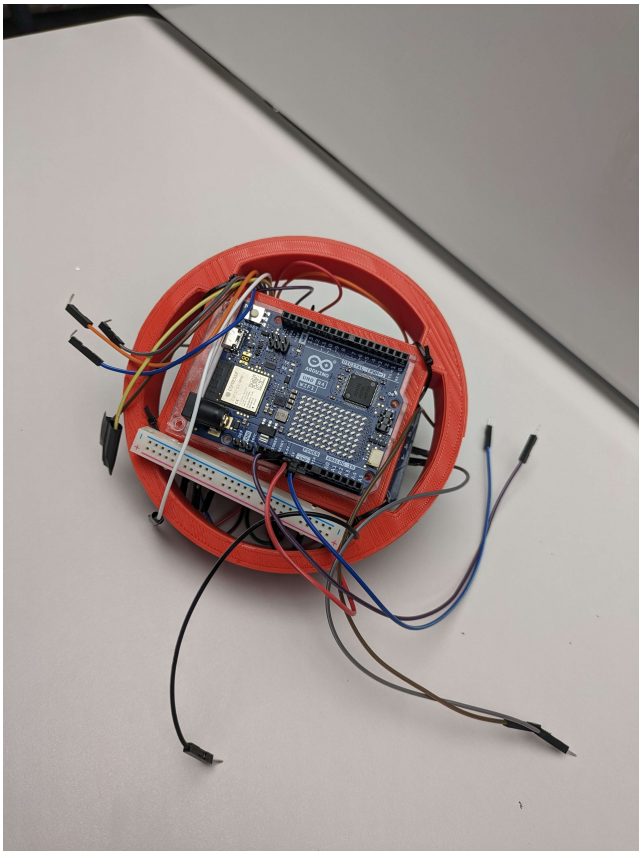


Figure 15: Electronics Module

Figure 16: Weighted Mesh

2.8 Testing Protocol and Results

Table 7: Test Plan

Team	Work Package	Requirement	Test	Test date
Systems & Control (A)	Buoyancy	A1.Change buoyancy of glider		
	Pitch Control	A2. Control Pitch angle of glider	A1a. Ensure engine takes in and expels water as required to control buoyancy	A1a. 19/3/24
			A1b. Ensure buoyancy works when manually controlled	A1b 25/3/24
			A1c. Ensure buoyancy works autonomously	A1c. 25/3/24
			A2a. Ensure pitch controller works in conjunction with IMU input	A2a. 29/3/24
Design & Build (B)	End Cap			
			B1a. Keep tube with end caps in a tub of water and ensure no leakages	B1a. 14/3/24
		B1. Seal Glider Tube	B2a. Ensure mesh fits into tube and holds components in place	B2a. 8/3/24
	Mesh Insert	B2. Hold components in place	B3a. Ensure Weights make glider neutrally buoyant	B3a. 5/4/24
	Weights	B3. Ensure neutral buoyancy		
Mechanical Systems (C)	Wing Clip			
		C1. Hold wings in place	C1a. Proper mounting	C1a. 18/2/24
	Nose	C2. Mount sensors and aerodynamics	C2a. Aerodynamic sim	C1b. 2/12/23
			C2b. Ensure sensors are mounted properly	C2a. 27/3/24
Communication (D)				
		D1. Communicate with Glider		D1a. 1/3/24
	Bluetooth Communication		D1a. Ensure a Bluetooth signal is received	D1b. 19/3/24
			D1b. Ensure Bluetooth can be used to send commands to glider	
Sensors & Recognition (E)	Ultrasonic sensor	E1. The sensor should be able to detect the obstacles and the lakebed in order to prevent damage to the glider		
			E1a. Ensure that the sensor measures the right distance when placed in a body of water	E1a. 25/3/24
	Temperature Sensor	E2. The temperature sensor should be able to accurately measure temperature of a water body	E1b. Ensure control loop moves from diving state to surfacing when input received	E1b. 25/3/24
		...		E2a. 25/3/24
			E2a. Ensure that sensor measures the right temperature when deployed	

Testing was carried out as described in table 7 to ensure functionality of all the subsystems. A more detailed description of important tests and their results are given in 2.8.1.

2.8.1 Testing Results

The bulk of the testing was carried out over 3 separate days. With 2 sets of tests being conducted in a controlled environment (tank) at UCL and 1 test being conducted at Mercer's Lake.

Sensor Testing

- Sonar Test 1 - Sonar placed in tank and calibration was checked to ensure it measures the distance accurately. The test was successful.
- Sonar Test 2 - Sonar was placed in tank and the controller was used to ensure that the control scheme moved from diving state to surfacing state accurately. The test was successful.
- Temperature Sensor Testing - The sensor was placed in the water along with a thermometer. Readings were then compared to ensure that the sensor was calibrated properly. The test was successful.

Glider Testing

This test was done to obtain neutral buoyancy. These tests made up the bulk of the testing as laying out the weights to ensure neutral buoyancy and static stability was harder than initially anticipated. Using the analysis performed in 2.2, it was known that the glider would have to weigh around 30kg to achieve neutral buoyancy.

- Weights were added to reach the desired weight of 30kg
- Extra weights were than added to the outside in increments of 500g and then 50g until neutral buoyancy was achieved
- Extra weights were than added to the outside in increments of 500g and then 50g until neutral buoyancy was achieved
- A final weight of 31.934kg was needed for neutral buoyancy

Hardware Testing

These tests were conducted to ensure that the hardware such as the linear actuators used in the operation of the glider functioned properly.

- Before being placed in the water the buoyancy engine and pitch control mechanism were controlled using a power supply to ensure that the mounting mechanisms were strong enough to withstand the actuation force of the linear actuators.
- Tests were also conducted outside the water to ensure that the actuators used for the buoyancy engine and could be controlled by the Arduino and powered by the battery pack.
- Finally the components were placed inside the glider and tested at both the lake and in the tank to ensure that all systems work as intended. The tests were successful.