OCE 496

Ocean Engineering Systems Design Project 2

STEPHEN LICHT, BRIDGET BUXTON

George Badlissi, Kaitlyn Barros, Joseph Grenier, Ian Hardman, Dan Josloff, Justin Lewis, Austin Myers, Yelena Randall

Project DigSki

11 May 2018

Contents

List of Figures

List of Tables

1 Abstract

The research vessel used in Sebastos Harbor by a group of underwater archaeologists led by Dr. Bridget Buxton is a major impediment in their ability to excavate due to its high cost, slow deployment time and poor suitability in coastal environments. A modified personal watercraft (PWC) able to power a venturi dredge resolves a typical research vessel's deficiencies while providing advanced capabilities in dredging and diver communication. In water testing was completed in order to prove this concept of PWC-operated dredging. Testing proved that it is possible to redirect and harness the outflow of the PWC but a PWC open system design prevented full scale dredge testing. Despite the open system nature of the PWC, the PWC's output volume flow rate was more than 230 GPM, which far exceeded expectations. A PWC-operated dredge system is viable but a closed system must be designed and dredging attachments must be able to function with high flow rates. A diver-operated PWC controller was implemented to ensure diver safety, and an auxiliary acoustic diver interface was created to provide an affordable solution for diver-to-diver communication and ranging.

2 Introduction

Current sub-bottom profiling technologies allow researchers to characterize the sea floor and identify buried artifacts by locating hard, dense buried masses. However, in environments where the sea floor is composed of dense reflective matter, these systems perform poorly. When this occurs, underwater archaeologists must turn to exploratory dredging to uncover artifacts at a given site.

Dredging is the act of excavating sediment, silt and other materials underwater. It is practiced on a large scale to widen waterways and shipping channels and also used on a smaller scale in shallow water (under 20m) environments by underwater archaeologists. Archaeologists use one of three small dredging systems – the air lift, the water jet and the water dredge.

The air lift, shown in Figure 1 is a deep water diver operated suction dredge. A topside mounted compressor pumps bursts of air down to the bottom of a vertical pipe submerged in the water. The air pocket formed rises to the surface. This creates suction in the pipe that pulls up water and sediment. Although this system is easy to manipulate on the seafloor, its topside components are large and bulky.

Figure 1: Picture of an airlift (GRD Franmarine)

The water jet is a shallow water diver operated jet dredge. A topside mounted water pump sends high velocity water down hosing, where a diver on the seafloor displaces sand using the outgoing jet. Although the water jet is cheaper than the air lift, it is not effective in displacing large amounts of sediment.

The water dredge, like the water jet, is a diver operated dredging system that uses a topside mounted water pump to send high velocity water down hosing. This hosing then feeds into one of the two inlets of a 'Y' pipe. The high velocity water causes suction to be formed at the 'Y' pipe's second inlet due to the creation of a low pressure zone at this point. This suction inlet is then used to suction debris off of the ocean floor. The debris and water are discharged from the pipe's outlet.

Because all three of these commonly used methods require the use of topside support, industry standards call for the use of a research vessel working in tandem with a two person dive team. The standard use of a research vessel as topside dredging support is a limiting factor in shallow water research. Research vessels are costly, unfit for dynamic shallow water environments and must be returned to a slip at the end of every work day- forcing long transits to and from a site of interest. These limitations prevail at the Sebastos Harbor (seen in Figure 2) archaeological dig site in Caesarea, Israel. In this particular area, artifacts are obscured by dense layers of coastal sediment, collapsed building stones and

bioencrustation that prevent archaeologists from using sub-bottom profiling technologies. This limitation forces them to displace the sand by dredging (Boyce et al. 2004). The site varies in water depth from 2m at its shallowest point to 10m (Boyce et al. 2004) at its deepest and experiences extreme swells of 2-3m (Buxton 2017). On a given weather permitting work day, the vessel, which bills at a rate of $1,600 per day, will depart from closest port and travel to the dredging site. Depending on the location of the site relative to the location of the closest port, the research vessel could be underway for hours before dredging operation begin. Once on site, the vessel will anchor – a daunting task for a large vessel whos anchor may damage artifcats. Divers begin dredging using a stern mounted pump. Once in the water, divers have limited hand signal communication with one another and little to no communication with topside. The divers, unbothered on the seafloor by any adverse weather, are able to dredge until they must resurface due to diving limitations. Because of the long transit time to and from a given site, the research vessel must stop operations if there is any likelihood of adverse weather in the near future. At the Sebastos Harbor location, transient conditions add urgency to this cultural heritage rescue. Because of its urgency, a dredging platform outfitted for the shallow water coastal environment has been developed using a PWC. Although urgency drove the re-design of the dredging platform at the Sebastos Harbor archaeological dig site , this PWC topside platform will be able to be used at any shallow water site.

Figure 2: Sebastos Harbor

3 Project Requirements

The system being designed must improve upon the current topside support standard in underwater archeology. Since present subsurface operations are deemed sufficient by archaeologists, the new system must dredge at least at the current rate of 2.5 m^3 of sediment per day. The system must also transport the dredge and at least two divers to and from the work site, which can range from 10-5000 meters offshore depending on the application. Safety is a priority when there are divers in the water and thus the system requires an emergency engine shut-off and the ability to transport an injured or incapacitated diver. The system should also have room for additional capabilities in the future. Examples of these capabilities are surveying equipment (i.e. side scan sonar) or robotic platform support for ROVs and AUVs.

4 Platform Justification

In order to validate the use of a PWC as a viable research platform, previous uses of a PWC were investigated. PWCs have been successfully used in nearshore environment research. One instance is the Coastal Profiling System (CPS), which is a PWC based platform created by Oregon State University graduate student Jamie Macmahan. It utilized a real-time global positioning system as well as an echosounder to obtain bathymetry data in the surf zone. In order to test the accuracy of the system, it was used at Myrtle Beach, South Carolina alongside the U.S. Geographical Survey. The first iteration of the system was unsuccessful because it did not meet accuracy standards of less than 5 cm vertically. The second iteration achieved acceptable levels of accuracy with an echosounder that operated at a higher frequency than that of the original iteration (MacMahan 2001). With the addition of a higher frequency echosounder, an acceptable bottom profile was achieved. This higher frequency combatted the unacceptable error produced by the large fluctuation in PWC depth in the coastal area. An image of the second iteration of the platform can be seen below in Figure 3.

Figure 3: Front View of the Coastal Profiling System(CPS): (A) Three-man Wave Runner; (B) ADCP Mount; (C) Stern Wave Runner Compartment Where GPS, Echosounder, and Laptop Computer Are Stored; (D) GPS Antenna; (E) External Screen Bellows; (F) External Screen Mounted in Watertight Case (MacMahan 2001)

Another PWC based platform was developed by the Areté Associates Washington Office. This platform was known as the Surf and Coastal Area Measurement Platform (SCAMP). It was tested at the US Army Corps of Engineers Field Research Facility at Duck, North Carolina and was able to take bathymetry data within the accepted error.

Figure 4: Personal Watercraft Instrumentation used at Rincón, Puerto Rico in 2012: (1) watertight laptop case, (2) Garmin gps reciever,(3) side scan sonar tow stand, (4) single beam echosounder,(5)Starfish side scan sonar, (6) solar panel, Chardon and Canals 2012

SCAMP (See Figure 4) was created to study the bathymetry of the surf zone near Rincón, Puerto Rico to determine the morphological response of the surf zone from 2012's Hurricane Isaac. A Starfish side scan sonar along with an echosounder and a real time kinematic (RTK) global positioning system were used to obtain and generate bathymetry data. SCAMP successfully documented cross-shore sediment transport after Hurricane Isaac (Chardon and Canals 2012).

The experiments detailed above are examples of successful cases where researchers used a PWC to collect data in the nearshore environment. The PWC was the chosen platform compared to a standard research vessel because it has greater mobility and the ability to access areas and launch from places a larger research vessel cannot. The PWC does not have the carrying capacity that a research vessel

would however, in the cases above the research teams were able to outfit the PWC with the tools needed to complete their tests. The equipment that is used in the previous cases is similar to the equipment that the PWC dredging platform would need to support.

After investigating other platform options, a PWC was chosen for the platform of the system. PWCs are less expensive and are more available than the commonly used research vessel and are also designed specifically for coastal areas like Sebastos Harbor. The inboard marine engine on the PWC also has the potential of powering a venturi dredge and can be operated solely by a diver, eliminating the previous need for hiring a costly captain and crew.

5 System Overview

The overall system design will be comprised of two subsystems. The first subsystem is the dredge system. The second subsystem is a diver interface subsystem that allows the diver to control the PWC and communicate with other divers and the topside operators from the subsurface.

5.1 Dredge Subsystem

The dredge subsystem allows for the PWC to be adapted as an underwater archaeological tool. The use of a PWC to replace the standard research vessel and hull-mounted pump allows for versatility in shallow waters and easy deployability from beaches and boat ramps. This in turn, reduces transit times and allows for more mobility and agility working in shallow water environments.

In order to use a PWC as a topside support platform, the inboard marine engine will be used to replace the current topside dredging pump. To use the PWC engine as a pump, the main components of the dredge subsystem consist of a redirection piece that redirects the outflow of the PWC down into a jet pump via fire hose. The second component of the dredge subsystem is the jet pump, this is what causes suction and thus the ability to dredge sand. This is shown in Figure 5

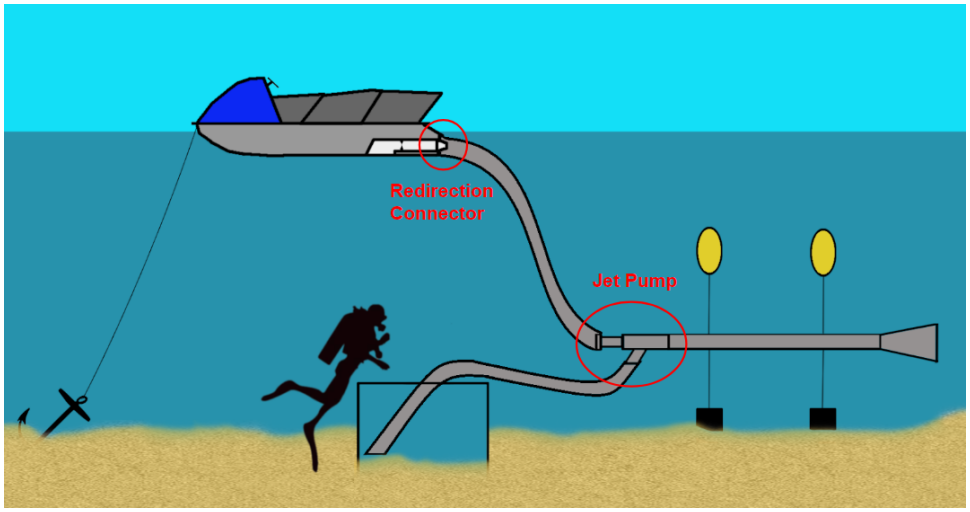


Figure 5: Dredge subsystem Overview

An auxiliary dredge subsystem is comprised of two parts; the PWC controller and diver watch interface. These systems, most importantly, increase the safety of the diver interacting with the dredge system and augment the capabilities of the system as a whole. This subsystem gives subsea ranging information, temperature and heading to the divers via the diver watch interface. The PWC control gives divers a safety emergency engine shut off and the ability to control the dredge system from below. The diver interface subsystem overview diagram can be seen below in Figure 32.

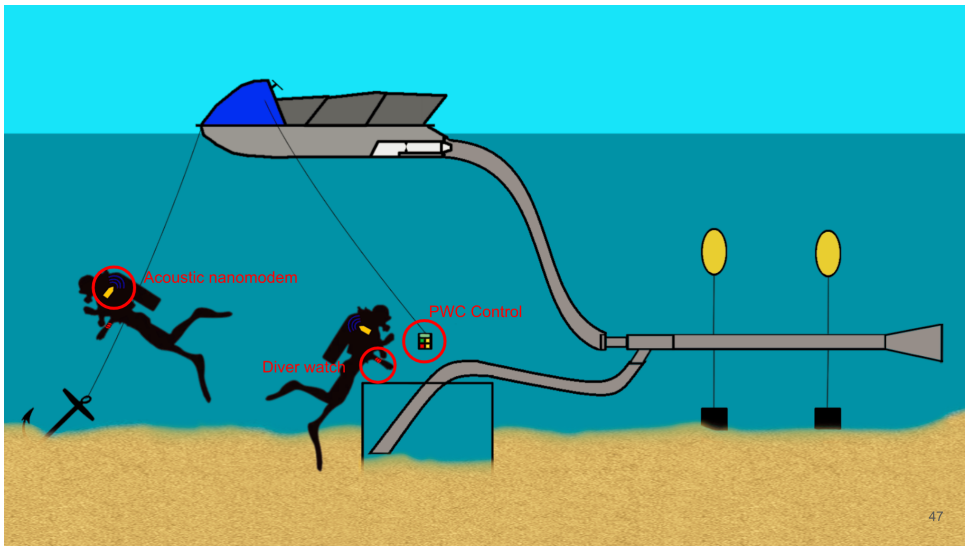


Figure 6: Diver Interface subsystem Overview

6 Dredge System Design

6.1 Jet Pump Theory

A liquid jet pump, or eductor, uses the energy of a primary fluid to induce the suction of a secondary fluid into the jet pump via a transfer of energy. This transfer of energy occurs in the four static components of the jet pump — the nozzle, throat entry, throat and diffuser, as seen in Figure 7. The primary fluid enters the jet pump via the inlet and continues through to the nozzle. At the nozzle the flow is restricted which causes a decrease in the fluid's pressure as potential energy is transfered into kinetic energy. The resultant high velocity jet exits into the throat entry and induces a low pressure zone around the nozzle. This triggers the induction of a secondary fluid through the suction inlet. The mixing of the primary and secondary fluids occurs in the throat where fluid pressure decreases. The diffuser transforms the kinetic energy of the mixture back to potential energy and follows the assumption that the mixed fluid discharges into the same pressure as the surrounding environment.

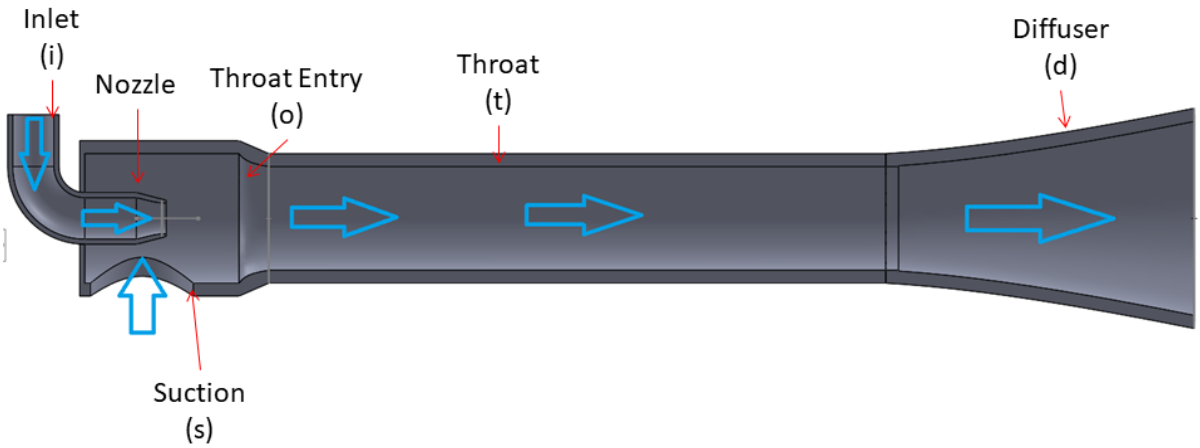


Figure 7: Jet Pump diagram showing four static parts: nozzle, throat entry, throat and diffuser. Also note sections labeled with parentheses, which denote subscripts used in the model equations (Karassik et al. 2001)

A jet pump model derived by Cunningham was used to design the jet pump (Karassik et al. 2001). The model was studied to fully understand the functionality of a dredge and will be used to determine the optimal dimensions of a jet pump operating at the given site. The following description of the jet pump design and model follows (Karassik et al. 2001). The model is based on conservation equations for mass, momentum and energy. The model assumes:

(a) Real-fluid losses accounted for by friction-loss coefficients (K).

(b) Friction-loss coefficients (K) remain constant for all operating conditions of pump.

(c) The primary and secondary streams enter the throat with uniform velocity distributions and the mixed flows leave the throat and diffuser with a uniform velocity profile.

(d) The primary and secondary fluids mixing process concludes in the throat before the diffuser.

(e) The mixed fluids discharge at the same pressure as the environment that the mixed fluids are discharging into.

(f) The water temperature remains constant.

(g) The flow is not cavitating.

For each of the four parts of the jet pump, there is a corresponding equation that expresses the change of pressure and other parameters throughout the pump:

$$\text{Nozzle: } P_i - P_o = Z\,(1 + K_n) \tag{1}$$

$$\text{Throat Entry: } P_s - P_o = ZS\,(1 + K_{en})\,\frac{M^2}{c^2} \tag{2}$$

$$\text{Throat: } P_t - P_o = Z[2b + \frac{2SM^2b^2}{1-b} - b^2(2 + K_{th})(1 + SM)(1 + M)] \tag{3}$$

$$\text{Diffuser: } P_d - P_t = Zb^2(1 + SM)(1 + M)(1 - K_{di} - a^2) \tag{4}$$

The nozzle tip experiences a discharge pressure close to P_s, not P_o, when the nozzle is retracted from the throat entry; thus, Equation 1 becomes:

$$P_i - P_s = Z\,(1 + K_n) \tag{5}$$

The pump efficiency is defined as the ratio of power output over power input:

$$\eta = \frac{Q_2\,(P_d - P_s)}{Q_1\,(P_i - P_d)} \tag{6}$$

The theoretical pressure characteristic, or pressure ratio, N, is defined as a ratio of pressure output over pressure input:

$$N = \frac{\eta}{M} = \frac{P_d - P_s}{P_i - P_d} \tag{7}$$

Combining equations 1-5, the theoretical pressure characteristic is:

$$N = \frac{2b + \frac{2SM^2b^2}{1-b} - b^2\left(1 + K_{td} + a^2\right)(1+M)(1+SM) - \frac{SM^2}{c^2}(1+K_{en})}{1 + K_n - numerator} \tag{8}$$

Another theoretical pressure characteristic form is:

$$\frac{P_d - P_s}{P_i - P_s} = \frac{N}{N+1} \tag{9}$$

The jet pump must operate in non-cavitating operating conditions (Karassik et al. 2001). The operational flow ratio of the pump must be less than the cavitation-limited flow ratio M_L to ensure that there is no cavitation.

$$M_L = c\sqrt{\frac{P_s - P_v}{\sigma Z}} \tag{10}$$

The liquid jet pump model only includes pump dimensions that are perpendicular to the flow. Longitudinal dimensions of the liquid jet pump must be obtained from the literature or determined experimentally.

Nozzle-throat spacing sp is recommended to be:

$$\frac{sp}{D_{th}} = 1 \tag{11}$$

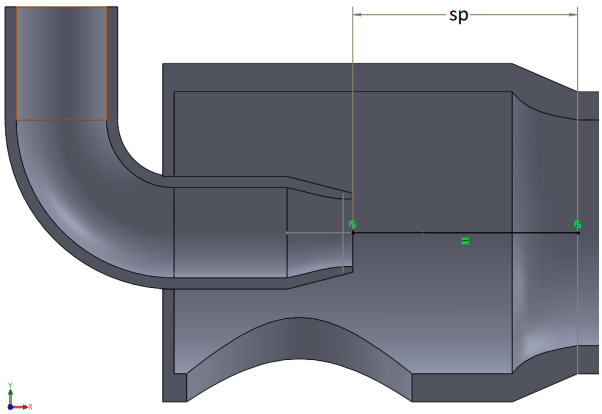


Figure 8: A diagram showing the nozzle-throat spacing distance, sp

Maximum efficiency η is achieved for $\frac{sp}{D_{th}} = 0$ but no spacing promotes cavitation. Retracting the nozzle a distance of about one diameter provides good cavitation resistance with only a small loss in efficiency.

Throat length L is recommended to be:

$$\frac{L}{D_{th}} = 6 \tag{12}$$

The throat should be long enough to allow the mixing process to conclude but as short as possible to minimize frictional losses.

Short-entry internally convex profile primary-flow nozzle design is recommended. Additionally, a short entry to the throat and a well-rounded profile connecting the suction chamber and throat is recommended. This is to minimize frictional losses and to maximize cavitation resistance.

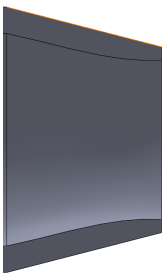


Figure 9: A two dimensional cross-section view of the entry of the nozzle, where it is short-entry and is internally concave

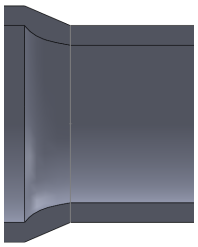


Figure 10: A two dimensional cross-section view of the throat entry, where it is short-entry and has a well-rounded profile

In order to convert the kinetic energy put into the system back into potential energy, the static diffuser is necessary. The angle of the diffuser, $Angle = Diffuser\ length\ \times\ tan(\frac{D_{di}-D_{th}}{2})$, is suggested to be between 5-8 degrees for optimal systems.

6.2 Jet Pump Design Methods

K friction-loss coefficients were adopted from published results from jet pump studies. The recommended K values are provided by Karrasik, et al. (2001):

Table 1: Recommended friction loss coefficients

K Coefficient	Recommended Values
K_n	0.05
K_{en}	0
K_{td}	0.20

The cavitation coefficient was also adopted from published studies. A conservative value of $\sigma = 1.35$ is recommended (Karassik et al. 2001).

The primary fluid density ρ_1 is the density of salt water:

$$\rho_1 = 1025\frac{kg}{m^3} \tag{13}$$

The secondary fluid density ρ_2 is the density of salt water and sand mixture, assuming that the sand

constitutes 25% of the mixture:

$$\rho_2 = 0.75\rho_1 + 0.25\rho_s \tag{14}$$

where sand density ρ_s is assumed to be the density of wet sand:

$$\rho_s = 1922\frac{kg}{m^3} \tag{15}$$

The density ratio S is computed:

$$S = \frac{\rho_2}{\rho_1} \tag{16}$$

The diffuser area ratio a is assumed to be small:

$$a = \sqrt{\frac{A_{th}}{A_d}} = \sqrt{\frac{D_{th}^2}{D_{di}^2}} \approx 0 \tag{17}$$

Picking a ratio of:

$$\frac{D_{th}}{D_{di}} = \frac{1}{2} \tag{18}$$

The diffuser area ratio $a = 0.0625$.

The design depth h is assumed to be $6m$.

The hydrostatic pressure P_h is calculated for the design depth where the pump is placed:

$$P_h = \rho_1 g h \tag{19}$$

The lift pressure P_l was calculated assuming a suction lift height h_l of 1m:

$$P_l = \rho_1 g h_l \tag{20}$$

The suction pressure P_s is calculated:

$$P_s = P_{atm} + P_h - P_l \tag{21}$$

The diffuser pressure P_d is calculated:

$$P_d = P_a + P_h \tag{22}$$

The vapor pressure P_v is assumed to be 3.49 KPa.

The theoretical pressure characteristic N is computed as a function of a range of area ratio b, a range of flow rate ratio M, as well as the constants the the density ratio S and friction loss coefficients K_n, K_{en}, K_{td}.

The pump efficiency is then computed from the theoretical pressure characteristic N:

$$\eta = MN \tag{23}$$

The maximum pump efficiency is found for each value of area ratio of nozzle over throat b. For each maximum pump efficiency, the corresponding operable flow ratio M_{op} and corresponding pressure ratio N values are obtained.

The operating primary, or motive, volume flow rate Q_1 is calculated using the operable flow ratio M_{op}:

$$Q_1 = \frac{Q_2}{M_{op}} \tag{24}$$

The inlet pressure P_i is calculated using Equation 9.

The dynamic pressure at the nozzle Z is calculated using the equation Equation 5.

The cavitation limit flow ratio M_L is calculated using Equation 10.

$$A_n = Q1 \times \sqrt{\frac{\rho_{pf}}{2\ Z}} \tag{25}$$

Equation 25 is used to to obtain the nozzle diameter with:

$$D_n = \sqrt{\frac{4 \times A_n}{\pi}} \tag{26}$$

This is the equation used to obtain the area of the throat:

$$A_{th} = \frac{A_n}{b} \tag{27}$$

Equation 27 is used to obtain throat diameter with:

$$D_{th} = \sqrt{\frac{4 \times A_{th}}{\pi}} \tag{28}$$

The diffuser diameter D_{di} was calculated using Equation 18.

The nozzle to throat spacing sp was calculated using equation Equation 11.

The throat length L was calculated using Equation 12.

6.3 Model Results & Discussion

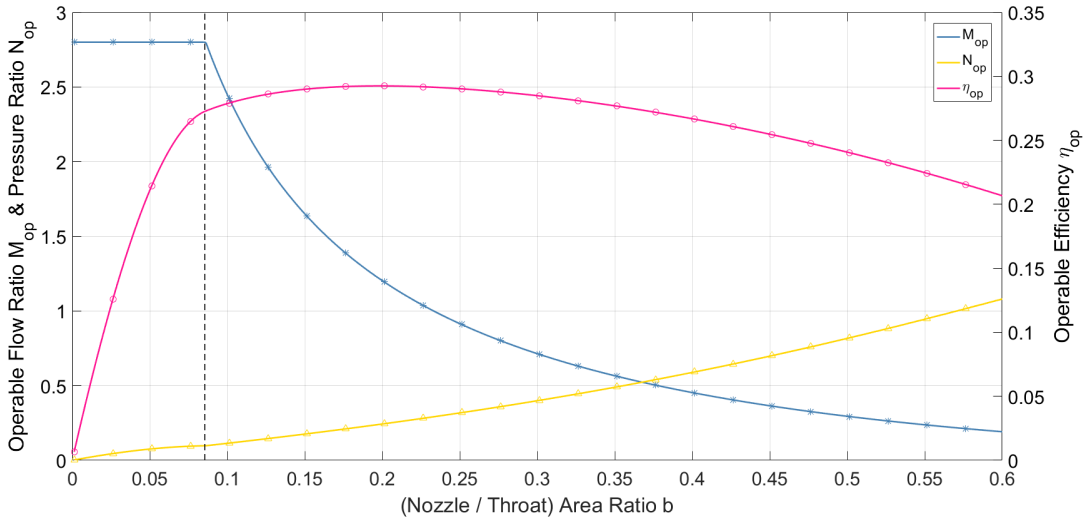


Figure 11: Plot of pressure ratio, N, flow ratio, M and efficiency over a range of area ratios, b

Figure 11 shows operable flow ratios and pressure ratios on the left y-axis, and operable efficiency on the right axis. Area ratios, b are along the x-axis. The area to the left of the vertical dashed line the model breaks down because large mixing losses develop, which are not accounted for in the model.

From this plot, a value of b with the greatest efficiency can be chosen, as long as the flow ratio, M, is less than 1. For this project, based on Figure 11, a b value of 0.25 was chosen, along with the associated M_{op} and N_{op}. This value was chosen for manufacturing reasons. Although it does not have the highest

efficiency, a ratio of the areas of 0.25 (1:4) means the area of the diameters will be 0.5 (1:2). These diameters are more likely to be available as stock sizes of pipes. This allows for easier acquisition. In addition, if the pump needs to be manufactured, any diameters that are not readily available need to be milled out. Milling has two negative impacts on the system. It introduces roughness in the pipe that will increase frictional losses.

After inputting the noted parameters into the model, the optimal dimensions for the jet pump were found. These are illustrated below in Figure 12. Labeled values are in meters.

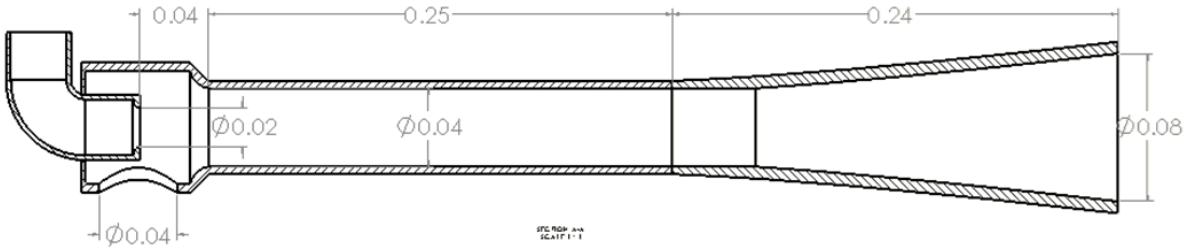


Figure 12: Optimal dimensions of the jet pump

These results are summarized in Table 2.

Table 2: Optimal dimensions of the jet pump

Dimension	Value
d_n	2.1 cm
d_t	4.2 cm
d_s	4.2 cm
d_d	8.4 cm
sp	4.2 cm
L_t	25 cm
L_d	24 cm

These dimensions lead to an operable flow ratio, M_{op}, of 0.918, which is less than the flow ratio limit, M_L, of 4.98. This means that the assumption that the flow is not cavitating is correct.

The model results give the theoretical optimal dimensions for the parameters that were inputted. However, lateral dimensions such as the diameter of the suction inlet is not provided and must be

assumed.

7 Proof Of Concept Testing

Proof of concept testing was performed in order to confirm that a successful connection could be made to redirect PWC flow into a dredging system and to confirm that an eductor designed using the jet pump model could successful dredge.

7.1 PWC Redirection Testing

PWC volumetric flow rate specifications were largely unknown when the PWC was acquired. Because of this, the outflow rate supplied by the PWC needed to be determined in order to quantify the amount of water that would be fed into the dredging eductor. In addition, the method of outflow redirection needed to be assessed. For this, field tests were performed without attaching any dredge components.

7.1.1 First Iteration Redirection System

In order to attach a redirection system to a PWC, a connector must be made. The PWC has two nozzles at its stern, the jet nozzle and the steering nozzle as shown in Figure 13. The jet nozzle is a fixed nozzle that the impeller outflow first goes through. The steering nozzle is downstream of the jet nozzle and is able to move along the horizontal axis to direct the flow towards port or starboard.

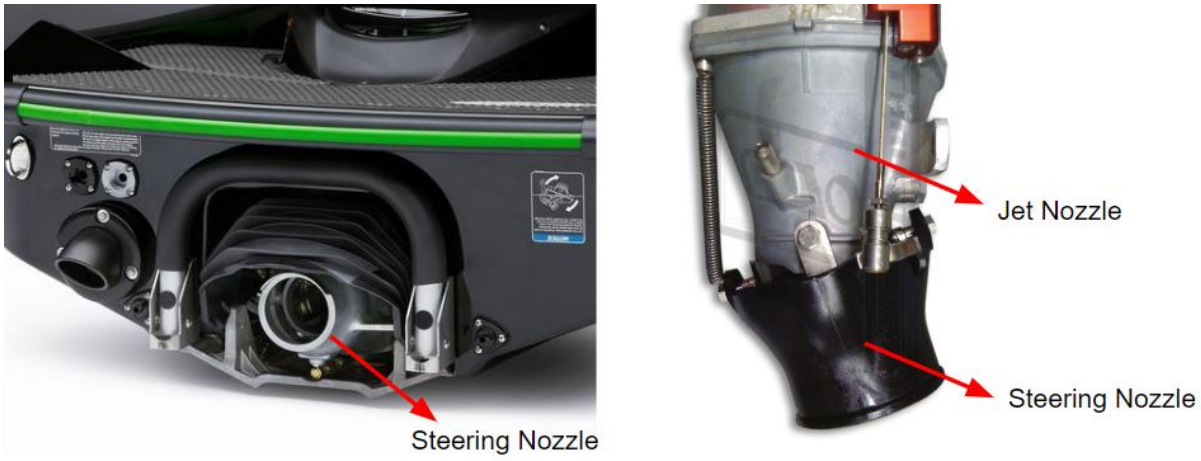


Figure 13: A diagram showing the PWC's steering nozzle and jet nozzle

The steering nozzle was modified so that a male threaded adapter could be attached. JB Weld was applied and three set screws were put in place to secure the threaded collar on. This modification can be seen below in Figure 14.

Figure 14: The modified steering nozzle with threading adapter

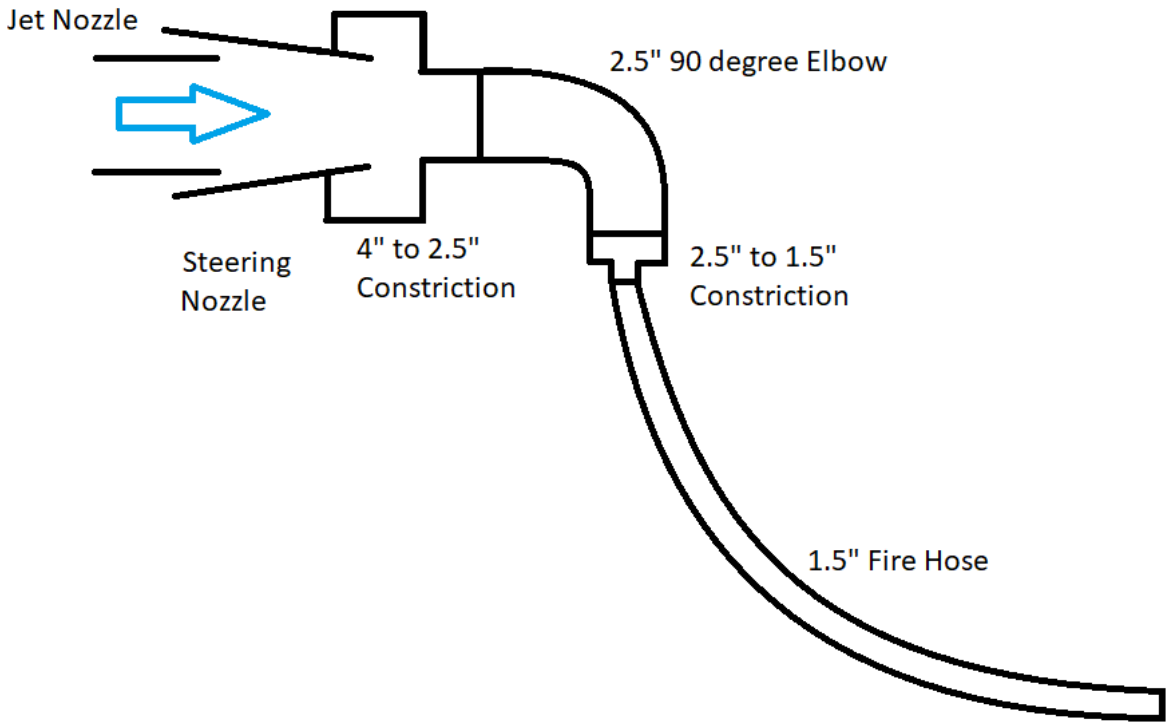


Figure 15: First iteration redirection system schematic

The first iteration redirection system was designed for the purpose of testing the PWC connection and flow redirection. PWC outflow is directed from the jet nozzle into the steering nozzle and through the redirection system which would in practice have jet pump connected aft of the fire hose.

The first iteration redirection system was made from threaded connections with multiple sudden, flat-plate reduction pieces that are often used on fire hose systems. These reductions are placed directly behind the jet and steering nozzle. A 90 degree elbow was then used to direct the flow down through a 50 foot 1.5 inch diameter fire hose and would feed into the eductor.

Field tests were conducted dockside at the University of Rhode Sailing Center. The tests were conducted during periods of high tide to ensure that the PWC was operating in at least 3 ft water depth, a standard minimum operating depth for PWC's.

In order to characterize the PWC outflow volumetric flow rate and understand the flow that would supply power to a future jet pump, the first iteration redirection system was connected to the PWC. The fire hose was run along the bottom and brought back up to the dock so that it would be discharging just over the edge. To quantify the volumetric flow rate, the discharge of the hose was directed into a

container of known volume. The time it took to fill the container was recorded for three RPMs on the PWC. The setup is seen below in Figure 16

Figure 16: Setup for testing PWC outflow volumetric flow rate

7.1.2 First Iteration Test Results and Discussion

The results of these tests are shown below in Table 3

Table 3: First Iteration Volume Flow Rates

Engine RPM	Volume Flow Rate (gal/min)
1700	27
2000	55
2500	128

During this testing, it was confirmed that it was possible to redirect the flow from the PWC. However, it is to be noted that there was turbulence at the PWC steering nozzle. This showed that not all of the water from the PWC outflow was entering the dredge system as the water naturally chose to follow a path of least resistance.

Figure 17: Comparison of the restricted and unrestricted outflows of the PWC

In Figure 17, the image on the left shows the system after the flat plate reductions have been attached. The bubbles show that there is flow that does not enter the system. The image on the right shows the unrestricted flow coming from the jet nozzle through the steering nozzle, the jet is fully formed.

The cause of this was determined to be the gap between the jet nozzle and the steering nozzle as shown in Figure 18.

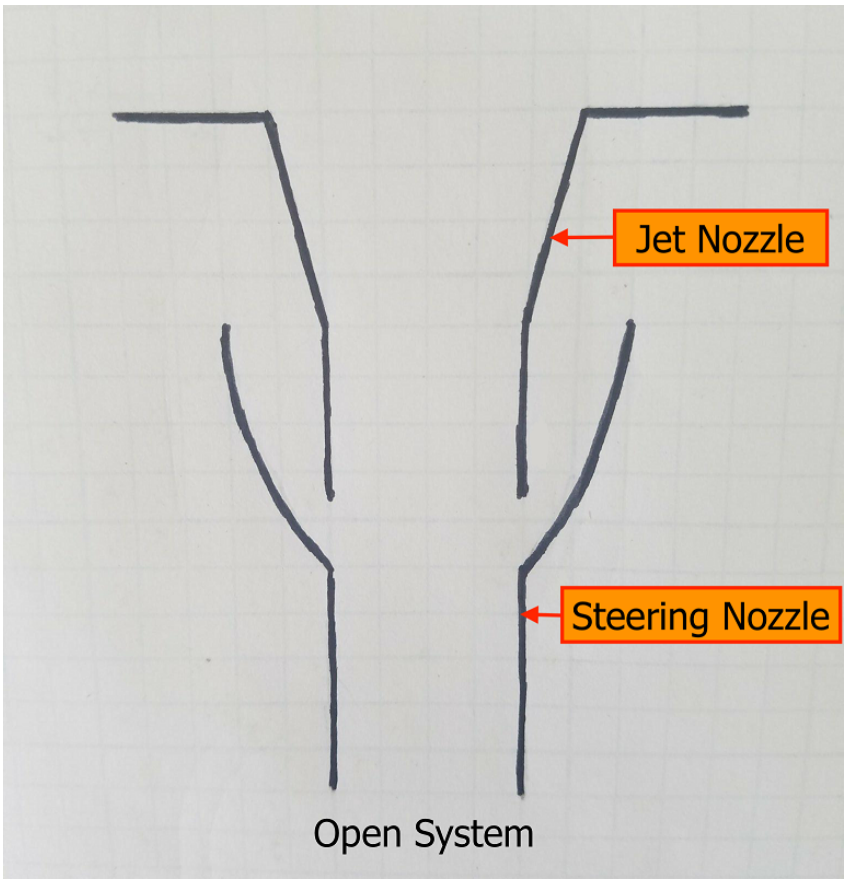


Figure 18: Cross section of the jet and steering nozzles

When the pressure was increased due to the reductions in the system, the flow was forced back out of the opening. This can be referred to as an the open system.

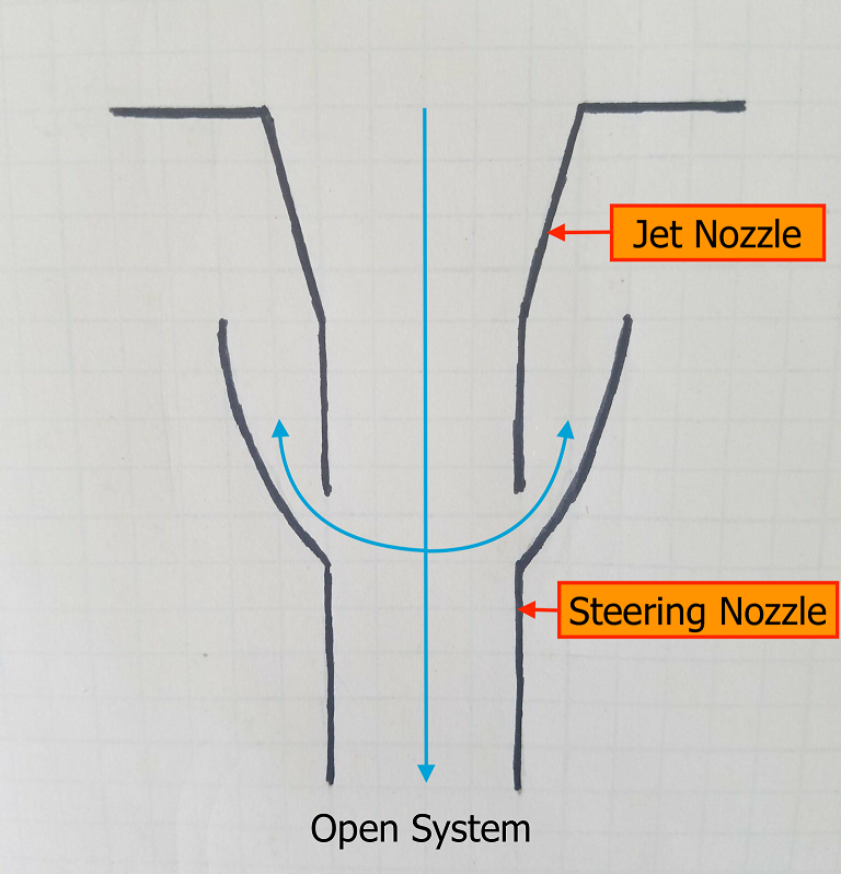


Figure 19: Cross section of the jet and steering nozzles with flow and forced backflow

Figure 19 shows the path the stream follows in this open system. Without any back pressure, the flow would go straight through from the jet nozzle out the steering nozzle. However, when the attachments are put on, flow is forced out the back of the steering nozzle, as shown by the blue arrows.

During testing, the threaded parts of the first iteration redirection system proved to be cumbersome to use and leaked at certain points.

7.1.3 Second Iteration System

Due to faults observed during testing with the first iteration redirection system, primarily the sudden contractions pieces causing increased pressure drop, a second iteration redirection system was designed. This system improved on the previous system's threaded connections and sudden flat plate reductions in an attempt to improve upon system functionality and ease of use.

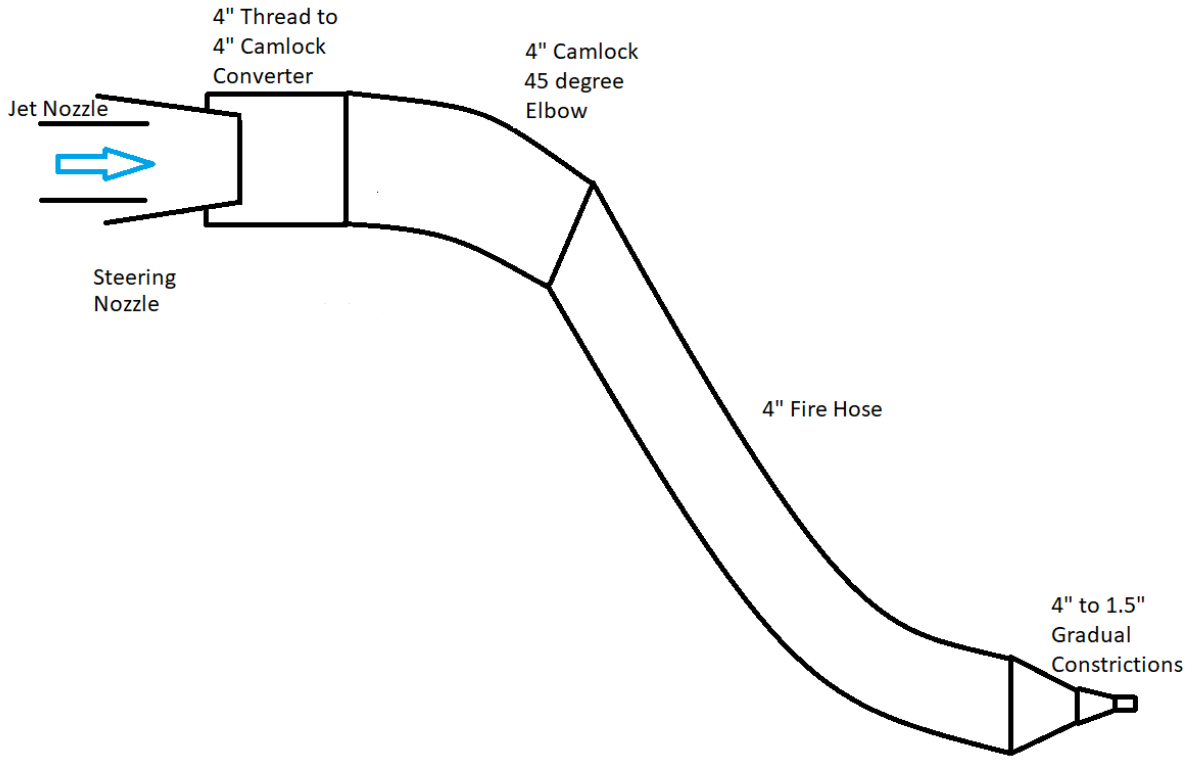


Figure 20: Second Iteration Redirection System Schematic

The second iteration system, outlined in Figure 20 features 45 degree gradual reductions aft of the fire hose as opposed to sudden flat plate eduction forward as outlined in the first iteration system. Cam lock and groove connections that provide a gasket water tight seal and are more user friendly then replaced treaded connections. In order to attach a new cam lock and groove based system to the PWC, threading on the steering nozzle was utilized with a thread to cam lock converter.

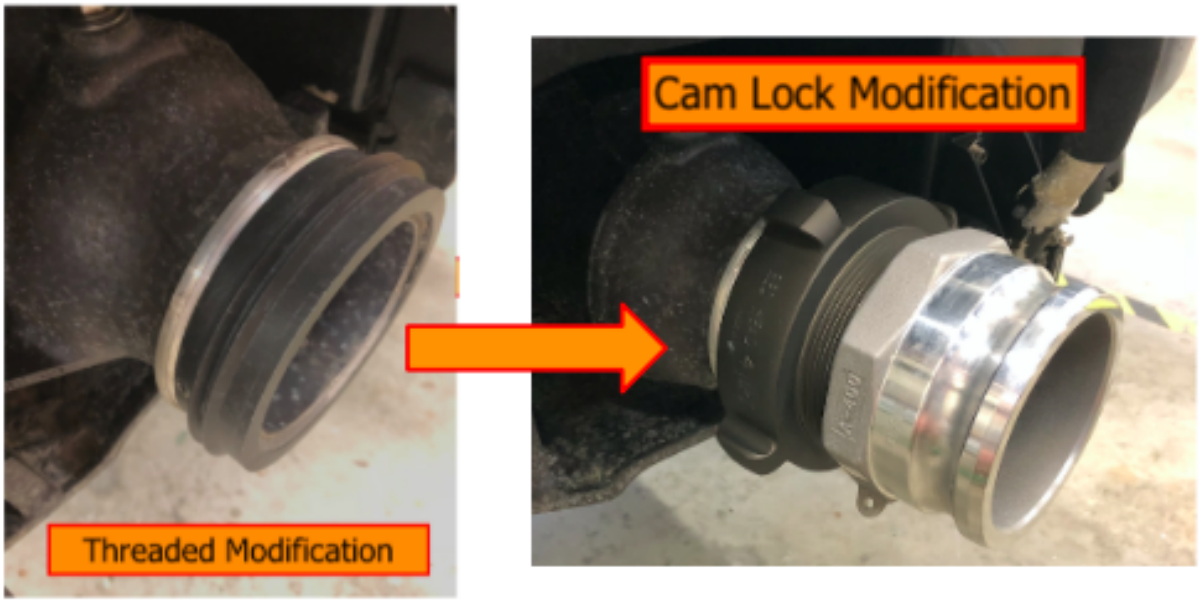


Figure 21: Second iteration Steering Nozzle Modification

The second iteration system also featured a simple cord stock o-ring designed in attempt to close the gap where water escaped during first iteration system testing between the jet and steering nozzle as seem in Figure 22. In such a figure, blue arrows again denote water flow.

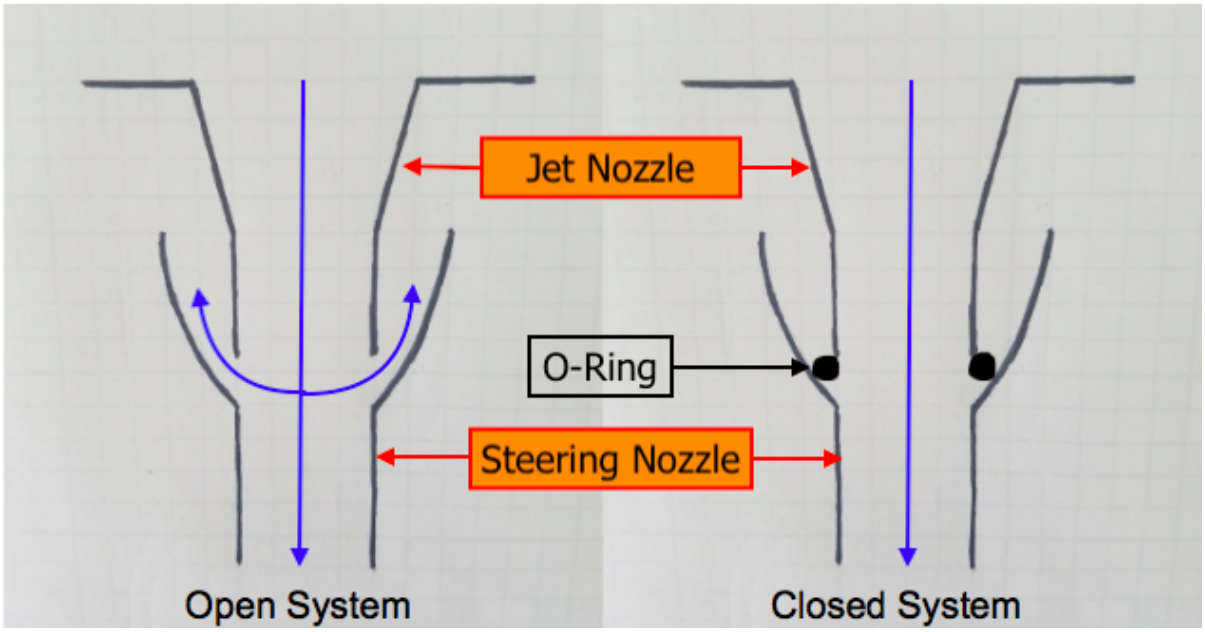


Figure 22: Closed System Modification

Because the o-ring was made from cord stock, the o-ring was imperfect. A small gap was observed and can be seen below in Figure 23 where the two faces of the stock meet.

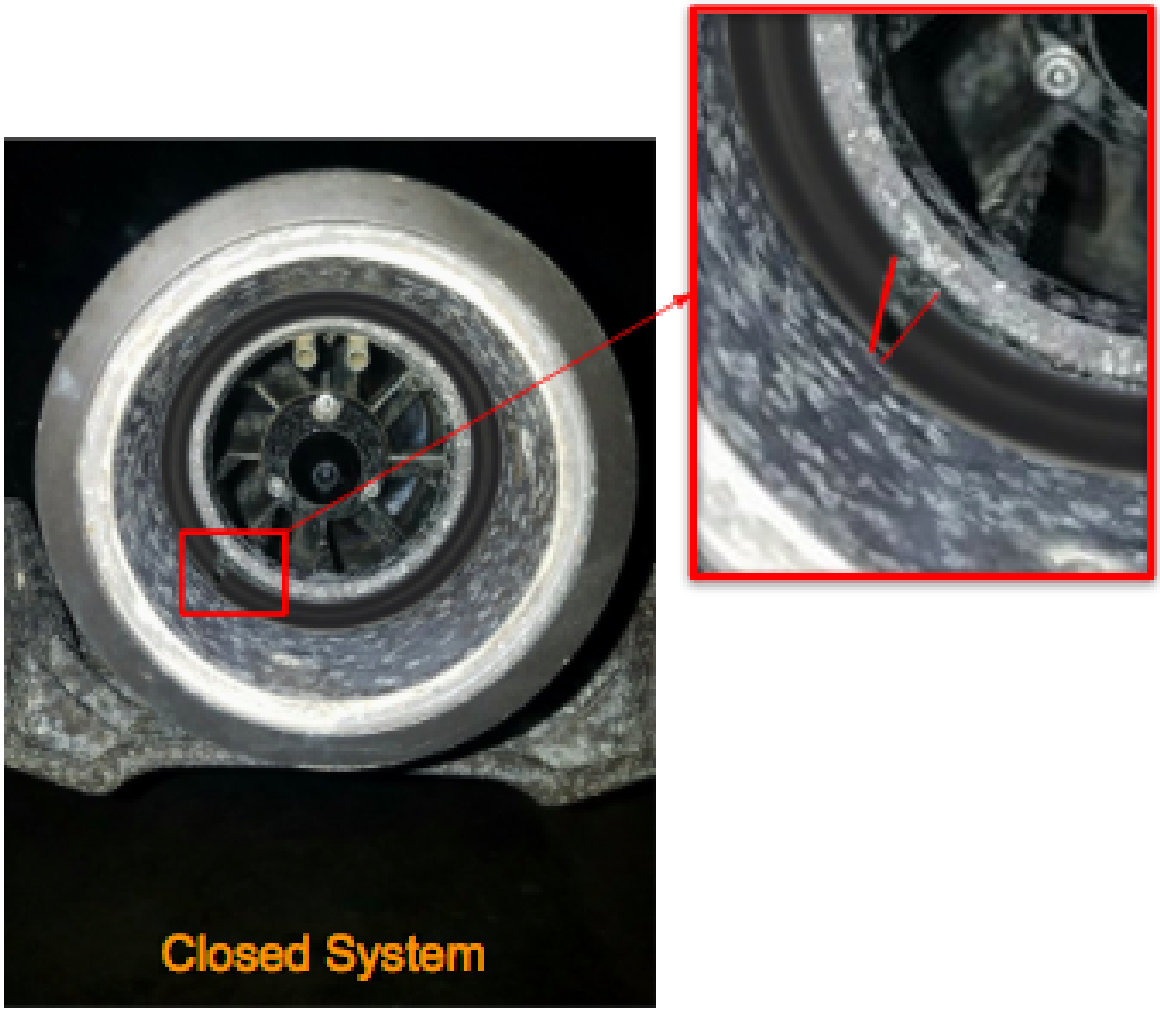


Figure 23: Closed System Modification

Testing methodology for the second iteration system mimicked that of the first system as both systems tested volumetric outflow rate and ease of use.

It should be noted that testing conducted with the o-ring installed will be referred to as closed system testing while testing conducted without the o-ring installed will be referred to as open system testing.

In addition, as seen in Figure 24, testing conducted with the cam lock converter, 4 inch fire hose, and gradual restrictions down to 1.5 inches will be referred to as constricted testing while testing done

with the cam lock converter and 4 inch fire hose will be referred to as unconstricted testing.

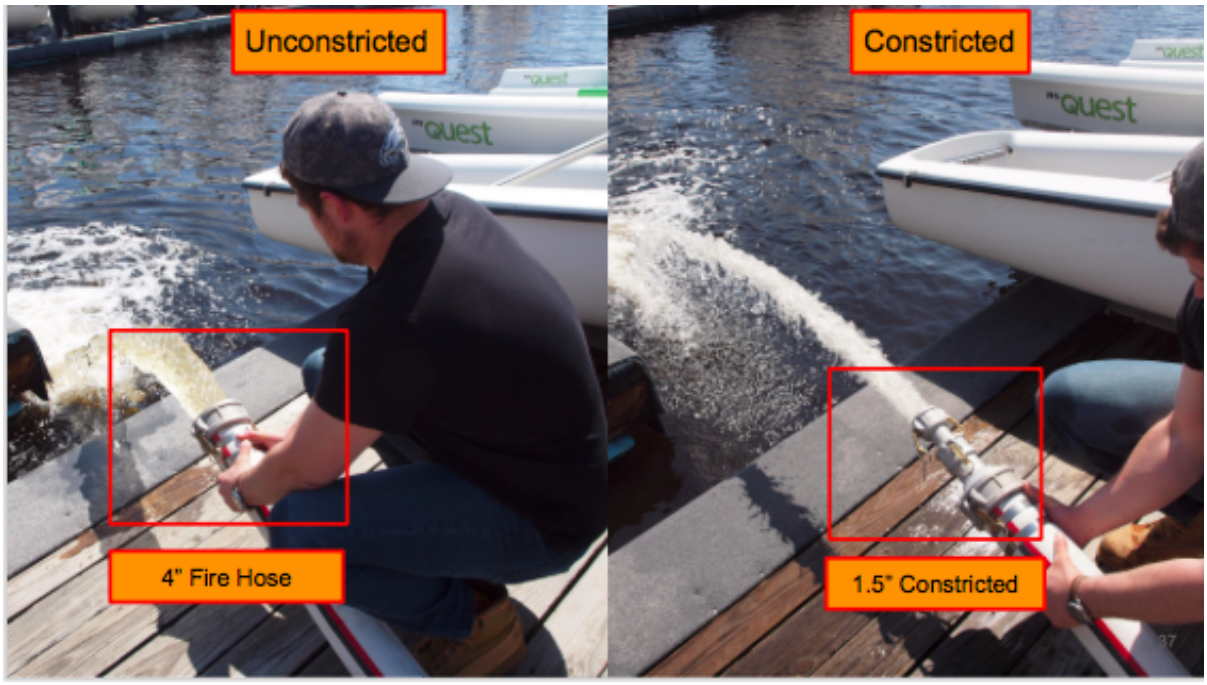


Figure 24: Constricted vs. Unconstricted Flow

For the system, volumetric flow rate was measured for both open and closed systems and with constricted and unconstricted flows for if a system was truly closed, the volumetric flow rate would be constant.

7.1.4 Second Iteration System Test Results & Discussion

Inconsistent volumetric flow rates were observed during closed system restricted and unrestricted testing. This means that the cord stock o-ring failed to close the system completely. Data for such testing is displayed in a table below.

Table 4: Volumetric Flow Rates

	Outflow Volume Flow Rate (gal / min)
Open System	81
Constricted Closed System	111
Unconstricted Closed System	229

Despite failing to close the dredging system using the cord stock o-ring, at idle, the volumetric flow rate was much greater than previously expected. If such a large volume flow rate was harnessed properly in a closed system, the PWC could power a large dredging system or possibly a dynamic multi-dredge system.

The initial thread to cam lock conversion piece was ran independently on the PWC in order to observe the nature of the output jet. When installed, the PWC jet was clean and resembled what the jet looked like without any attachments or modifications. This result was desired over the turbulent jet output that resulted when the initial first iteration flat plate reduction was ran on the PWC. Images of both normal PWC outflow and outflow with the attached converter can be seen below.

Figure 25: Normal PWC Outflow

Figure 26: Cam Lock Converter PWC Outflow

It is to be noted that the cam lock and groove system proved to be easier to use in this setting and successfully provided a water tight seal throughout the system.

7.2 Jet Pump Testing

The main goal of the jet pump testing was to confirm that a jet pump designed using the jet pump model was able to dredge sand at a predicted rate in order to confirm model accuracy and understanding. A jet pump theoretically eight times more efficient than the sponsors current pump (able to dredge hourly what the current system dredged daily) was designed from dimensions calculated using the pump model.

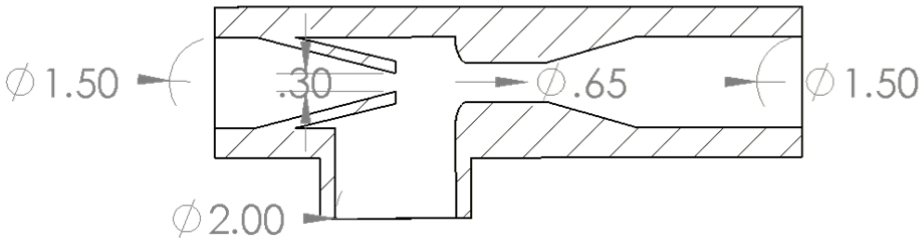


Figure 27: Cross section of the purchased commercial jet pump

Due to time constrictions and funding, a manufactured jet pump that was thought to dimensionally resemble the model designed jet pump was purchased. It is to be noted that after testing concluded, it was determined that the manufactured jet pump had a nozzle that was 63% smaller than expected.

Because PWC testing with redirection system showed dynamic fluctuating volumetric flow rates as water escaped through the jet and steering nozzle opening, jet pump testing was completed on an external pump. This allowed known constant volumetric flow rates to be supplied the jet pump.

Before jet pump testing could take place, the outflow of the external pump was characterized. In order to do such a thing, the external pump's suction hose was placed into the seawater and the second iteration redirection system was attached to the external pump's discharge. Volumetric flow rate was calculated for five pump power levels by filling a known volume in a measure timed.

In order operationally test the manufactured commercial jet pump, the jet pump was connected to the second iteration redirection system and powered by an external pump. The setup of the tests that measured the volume flow rate of the suction from the jet pump is shown below Figure 28.

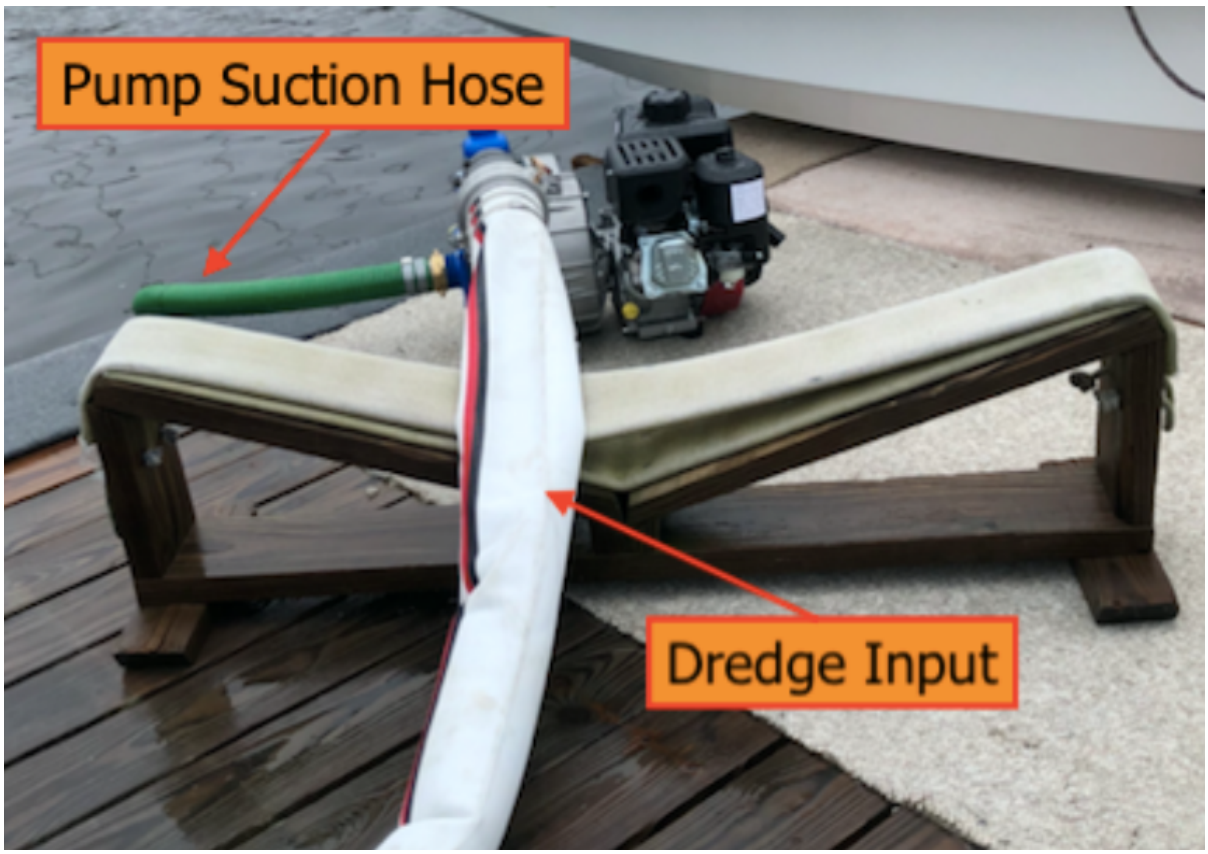


Figure 28: Setup with second iteration redirection system attached.

To validate the jet pump's ability to dredge, the jet pump suction volume flow rate was measured with both water to confirm basic functionality and a sand and water mixture to quantify the jet pumps dredging ability. The suction hose of the jet pump was operated inside a container which contained at

first sea water and then a mixture of water and sand. The discharge hose was held above a sieve that would allow water to go through but trap the discharged sand. The person operating the suction hose would agitate the sand with their other hand and observe the amount of sand being dredged. A timer was started once sand was observed coming out of the discharge house and onto the sieve. The timer was stopped when the suction container was emptied. The sand volume dredged would then be measured using a measuring cup. Images of the test are shown in Figure 29.

Figure 29: Left: suction hose sucking sand and water from the suction container. The operator is agitating the sand to ensure optimal suction. Right: discharge hose discharging sand and water into sieve.

7.2.1 Jet Pump Testing Results & Discussion

Testing the external pumps flow rate in relation to its power output concluded that at idle the input flow rate was 45 gallons per minute. The desired operable volume flow rate was 53 gallons per minute based on the model. However, due to system pressure concerns, all future testing was conducted at idle input flow rate of 45 gallons per minute. That idle flow rate of 45 gallons per minute is represented by the lowest data point in Figure 30.

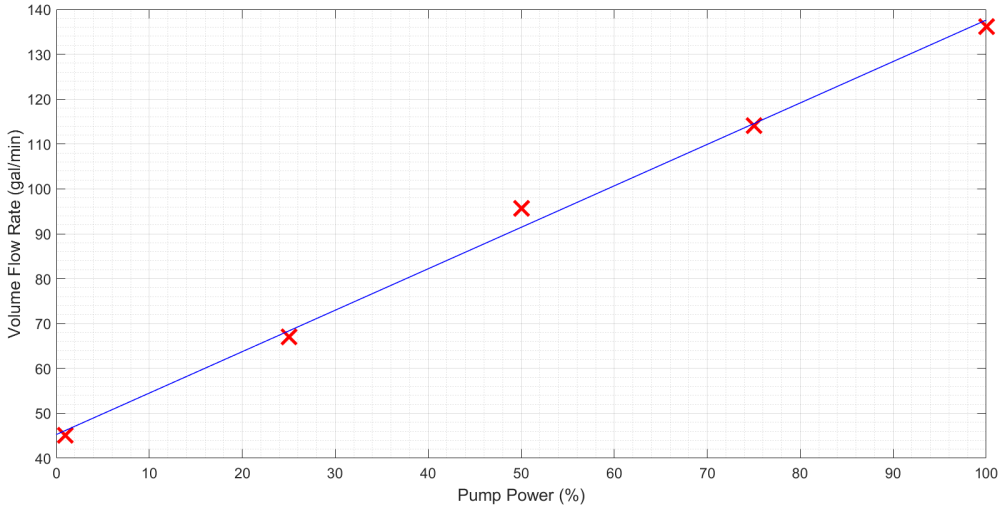


Figure 30: Volumetric Flow Rate of the External Pump for Various Power Levels

When the suction fluid was only seawater the jet pump yielded a 23 gallons per minute suction volumetric flow rate. When the test was run again but with the suction fluid being a mixture of sand and seawater, the jet pump yielded a flow rate of 3 gallons of sand per minute. The difference in density between the seawater and the sand mixture is very large and this is the reason that the flow rate appears to drop significantly. The other discrepancy in the data is that the sand and seawater mixture was only measured in gallons of sand as the seawater drained through the sieve. However, while being supplied 8 gallons per minute less than desired and while operating with a 63% smaller than desired nozzle, the jet pump dredging rate of 3 gallons of sand per minute achieves what the sponsor does per day in only 3 hours.

Table 5: Field Test Flow Rate Results

Input Q_1	45 gal/min
Avg. Suction Q_2 (Water only)	23 gal/min
Avg. Suction (Sand only)	3 gal/min

The jet pump was also placed on the PWC despite being unable to close the redirection system behind the steering nozzle. The jet pump setup was the same with the PWC as with the external pump. There was little discharge of sand seen. However, small amounts of sand could be seen being sucked into the jet pump as seen in Figure 31. This testing, although not quantitatively meaningful, allowed the dredge to operate on PWC power, further proving the concept at hand.

Figure 31: Suction of Sand Using PWC

7.3 Moving Forward

In order to further the success of this effort, it is suggested that the redirection system be modified to close the gap between the steering and jet nozzle. This will insure that the jet pump will be supplied an optimal constant volumetric flow rate. When this faster than expect flow gets to the jet pump, the current jet pump is not large enough to withstand the pressure of the flow. For this reason, it is suggested that the jet pump be resized in order to work at such high flow rates.

To improve upon the system at hand, it is also recommended that a pressure relief valve should be installed into the system and swivel connector should be installed to improve upon the system ease of use and prevent twisting and kinking while in use.

8 Diver Technology

8.1 Existing Diver Technology

Current dive technology allows for a diver to locate themselves spatially, communicate with other divers, and track their time subsurface. This technology comes at a significant cost though as popular dive computers can cost upwards of $200. For example, at such a rate, the Suunto Zoop Novo Dive Computer Wristwatch provides the most basic abilities such as depth, dive time, and temperature. The iDive underwater housing system in use at the Sebastos Harbor site is an alternative to diver specific hardware, and gives the divers the ability to use their iPads, with dive software preloaded, on the seafloor. This is a pressure regulated system that utilizes the diver's air tank. It does not provide communication or positioning, but allows for more detailed note taking, photo operations, and the use of preloaded maps to aid in subsea navigation. The iDive housing itself costs around $1000 and limiting in the sense that only certain hardware can be used. Systems such as the Shark Marine Technologies Navigator gives divers the ability to communicate with other divers, utilize diver position & location, and map habitats amongst other more specialized options offered (*Navigator: Diver Held Sonar Imaging and Navigation System* n.d.). However, this system costs upwards of $50,000. Although current dive options offer diverse capabilities, the cost and versatility of such products greatly limits those who can access these systems in underwater science. Because of this, the DigSki will utilize custom dive technology in the diver interface subsystem.

8.2 Diver Interface System

The diver interface system is comprised of two parts; the PWC controller and diver watch interface. These subsystems, most importantly, increase the safety of the diver interacting with the dredge system and augment the capabilities of the system as a whole. The diver watch gives subsea ranging information, temperature and heading to the divers. The PWC control gives divers an emergency engine shut off and the ability to control the PWC throttle from below. The diver interface system overview diagram can be seen below in Figure 32.

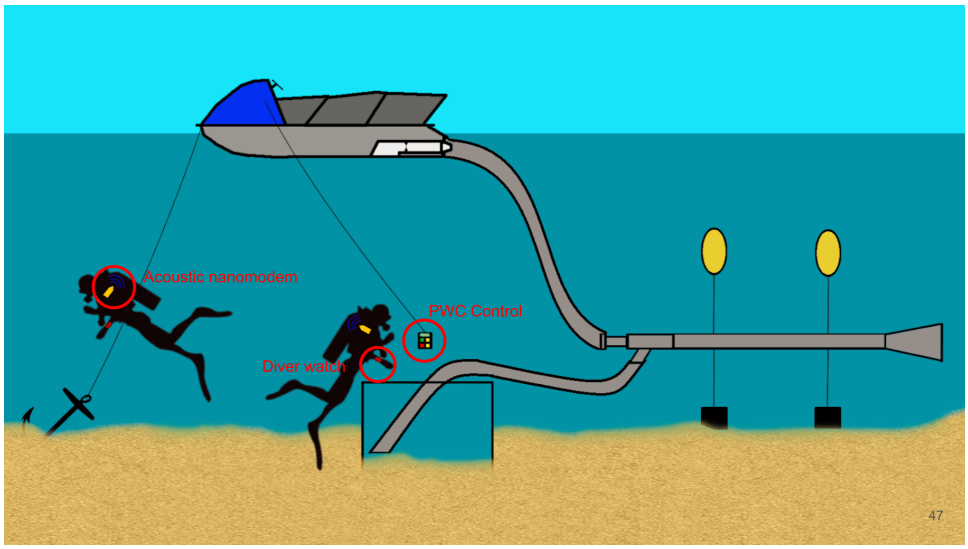


Figure 32: Diver Interface Module Overview

8.3 Subsystem 1: Dredge Control

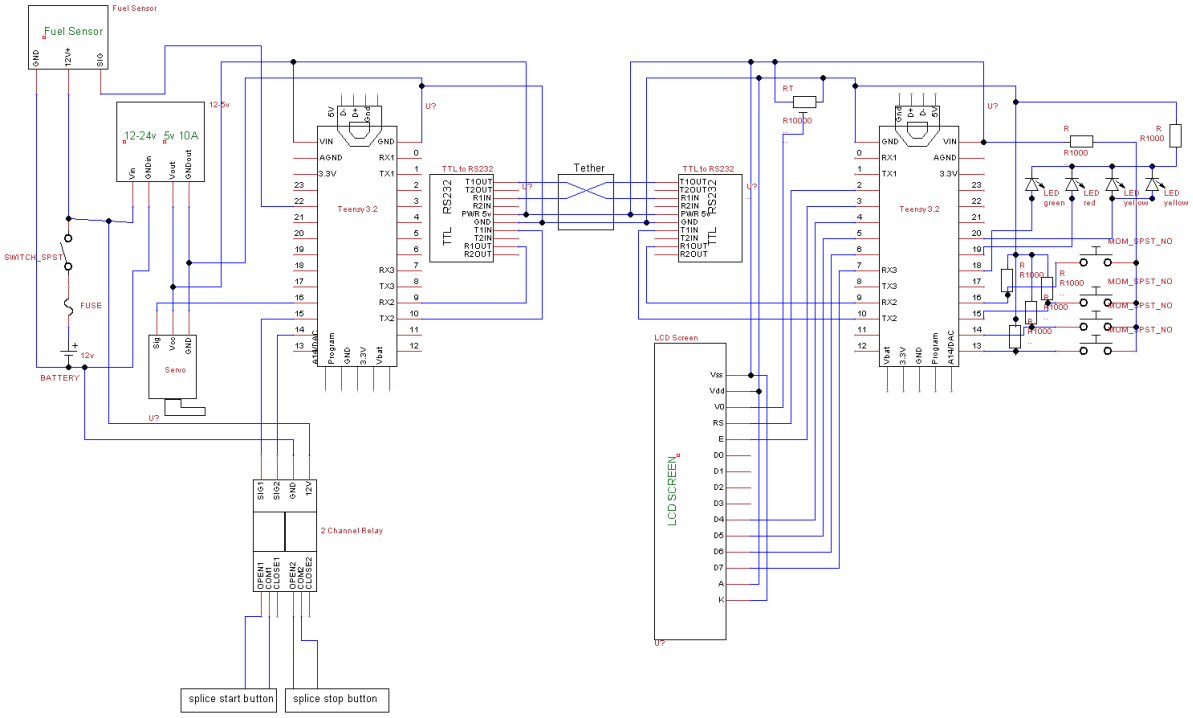


Figure 33: Dredge Control Wiring Diagram

The dredge control system is a tethered unit aimed to give the divers the ability to control the PWC from the ocean floor. The system consists of two components - a controller near the dredge on the seafloor and a topside PWC control system. A wiring diagram for the dredge controller can be seen in Figure 33. The controller is comprised of a liquid crystal display (LCD) screen to display information, four buttons to turn the PWC on or off, as well as increase or decrease the output of the engine and a microcontroller. This subsurface microcontroller is connected to a TTL to RS-232 converter to allow communication with a secondary microcontroller within the PWC on the surface over a 50ft tether.

The PWC microcontroller is connected to 2 relays, one to start the PWC engine and another to stop it. These will be connected in parallel to the engine start and stop switches found on the left handlebar of the PWC. Both the start and stop switches are normally open. For the dredge control system to start and stop the PWC by either the buttons on the handle bars, or the dredge controller, the relays will

need to be installed in parallel. To start the PWC, the start relay is turned on briefly as to not burn out the starter. Currently the start relay is turned on for 5 seconds and then turned off. Future iterations of the control unit will turn on the starter and read a pin connected to the output of the alternator to check whether or not the PWC is running. Once the microcontroller confirms the engine is on, it will then turn off the start relay.

In order for the dredge controller to adjust engine output, a secondary throttle cable must be attached to the existing throttle cable. This cable will be attached to a motor or servo to adjust the length of the cable, increasing or decreasing the engine output. A mechanical limiter will be placed on the motor so that it cannot run at full PWC engine strength while the dredge is connected. This will prevent damage to the dredge or divers, as the full strength of a PWC produces high pressures that can damage the dredging system. This limiter will not affect the top speed of the PWC while the PWC is being used for transport, as the throttle control cable will be mounted in parallel to the throttle cable on the handlebar.

The PWC control system will also be able to read the fuel gauge mounted on the PWC. The Teensy controller on topside will read an analog output of the gauge and relay this information to the control panel along with the state of the PWC (on or off). Additionally, engine speed and engine status will also be relayed down to the dredge controller and displayed on the LCD screen. This information will be updated each time there is a change. The dredge control will also be programmed so that if communication is lost between the controller and PWC, the engine will be cut. This assures that the divers will always have a way to stop the engine of the PWC. However, an override switch will allow the diver to turn off the PWC control system so that the engine can still be started for transport without the dredge controller. The current dredge control system can be seen in Figure 34. The topside controller can be seen out of its housing on the left, with attached relays and servo motor. The control panel is on the right hand side showing simulated PWC status of being on at 33% power of the motor. Both the control panel and PWC controller were built on proto-boards and were able to start a PWC during testing.

The dredge control system increases the safety of the system by allowing the divers to stop the engine while underwater in case of an emergency or to clear out the dredge and avoid surfacing while the PWC engine is running. The system will also allow the diver to increase or decrease the output of the PWC engine, which directly relates to the suction force of the dredge.

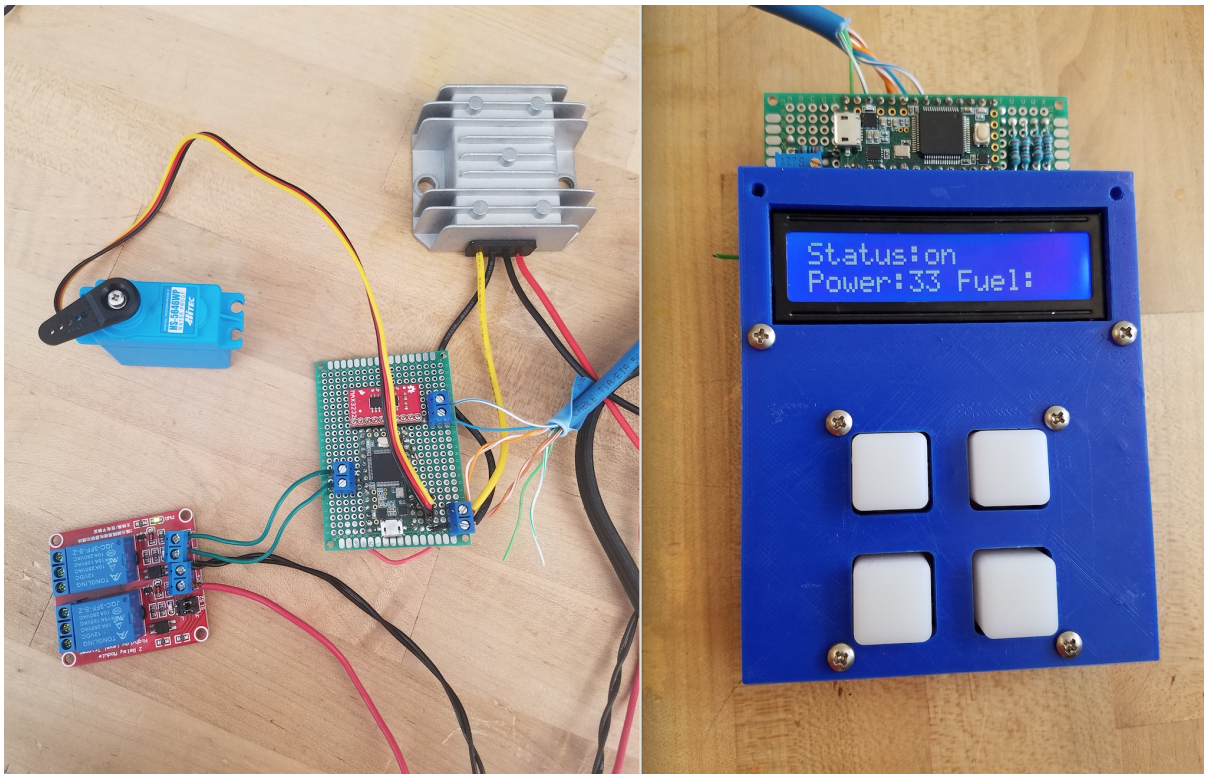


Figure 34: Dredge Control System

8.4 Moving Forward

Moving forward with the project, The topside and controller should have PCB boards printed to replace the protoboards currently in use. further testing and integration must also be done on other PWCs to test compatibility. A servo mount must also be designed for the throttle cable for implementation into the PWC as well.

8.5 Subsystem 2: Diver Watch

The diver watch is where the diver can interact with the acoustic modem and on-board sensor information. The full system overview is seen in Figure 35.

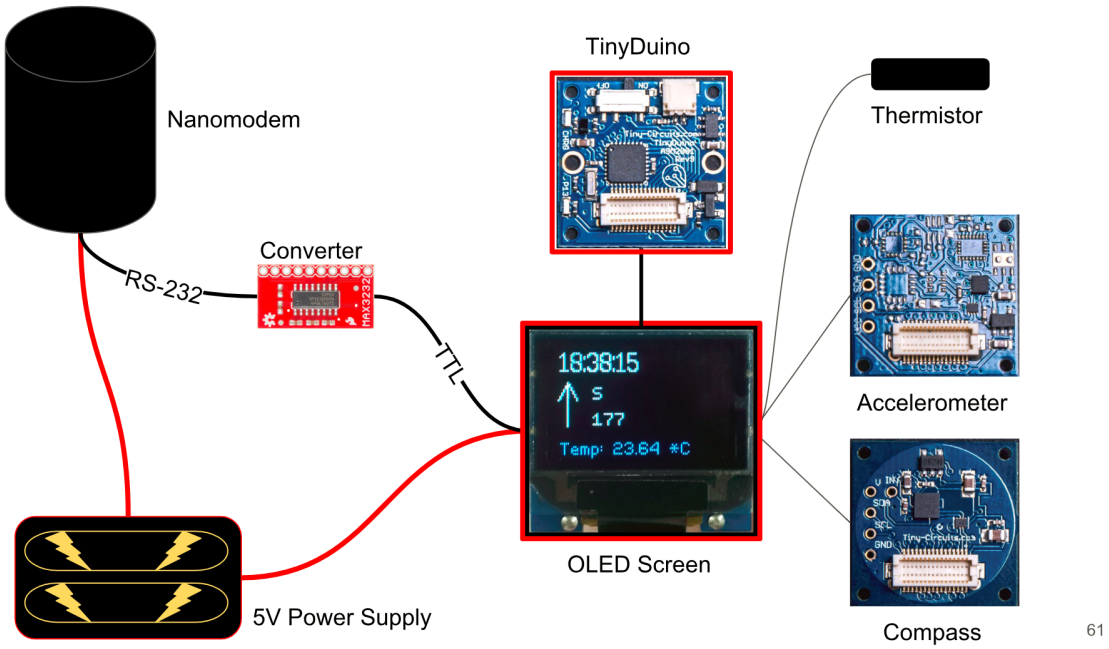


Figure 35: Diver Watch Interface System Overview

The nanomodem, converter, and TinyDuino (Arduino) microprocessor are all supplied by a single 5V power supply. The diver watch communicates with the nanomodem over TTL which is converted to RS232. The OLED Screen has four buttons, two of which are being used to interact with the watch in "boot-up" mode where the unit ID of the nanomodem that will be communicated with is selected, the time is set and instructions for a "hard-iron" compass calibration (Honeywell n.d.) are given. Those same two buttons are also used in the main loop in order to send the "ping" command which returns the range in meters from the other diver interface system. The compass heading shows a magnetic-North referenced heading angle and is tilt-corrected in order to account for small pitch and roll angles measured by the accelerometer that are inherent with a device that is mounted on a moving diver. The thermistor is a generic 10K waterproof probe, which measures resistance. This resistance is than converted to temperature in Celsius using the Steinhart-Hart equation (26).

$$T = \frac{1}{A + B * ln(Rt) + C * (ln(Rt))^3} \tag{29}$$

Where Rt is the thermistor resistance, and A, B and C are calculated using a three-point calibration. For this particular thermistor, A, B and C are calculated to be:

$$A = 3.412026533 \times 10^{-3} \qquad B = -0.93568566 \times 10^{-4} \qquad C = 11.49459621 \times 10^{-7}$$

The TinyCircuit shields are stacked below the OLED screen, the nanomodem is connected via "TX" and "RX" pins and the thermistor is connected via two anolog pins ($A0$ and $A2$).

8.6 Acoustic Transducers

In an effort to provide the diver with mobility without sacrificing their connectivity, an acoustic modem was deemed the most effective solution.

The final iteration of the diver watch uses Nanomodems made by Jeff Neasham at the Newcastle University (Figure 36) to provide ranging and communication abilities over up to 2 kilometers. The nanomodems cost about $50 to produce, and are small enough to be mounted on divers, underwater vehicles, and subsea instruments (60mm X 42mm). "Short data messages may be exchanged between units and an efficient 'ping' protocol is implemented for range measurement by transponder operation." (Neasham 2017). The diver as it stands uses two of the possible modem commands: "$?" to query the nanomodem status as well as "$Pxxx" to ping the modem with unit address xxx (see Appendix for more information).

Figure 36: Jeff Neasham, Newcastle University Nanomodem

Initially, modems were chosen for this project from Blueprint SubSea; the X150 modem beacon (seen in Figure 37) and the x010 modem. These modems allow for positioning and data transfer at ranges up to 1 kilometer, as well as depth information, water temperature , pitch, roll and yaw with

incorporated gyroscope, accelerometer, magnetometer, pressure and temperature sensors (Barratt and Sharphouse 2017). These modems were chosen initially due to a specific funding opportunity that never was received.

Figure 37: X150 USBL Transponder Beacon

Acoustic modems are subject to limitations. These limitations are the possibility of attenuation in the acoustic pulses that are sent, the maximum message size and the time to send messages. Message corruption can be caused by disturbances in the water such as diver bubbles or suspended sediment. Interference can also be caused by noise from the PWC. However, the noise from the PWC was deemed to have negligible effects on the acoustic transponders after discussions with Jeff Neasham of Newcastle University and Robin P. Sharphouse, co-director of Blueprint SubSea.

8.7 TinyCircuits

The final diver interface uses a mini Arduino microprocessor called TinyDuino made by TinyCircuits to run a simple diver interface that doesn't require a desktop environment. This option provides all necessary functions, is less complicated by nature and smaller in size. In this interface standard Arduino and TinyScreen libraries are used to interface with the OLED screen and on-board sensors. The SoftwareSerial library is used to communicate over the serial "RX" and "TX" pins on the TinyDuino.

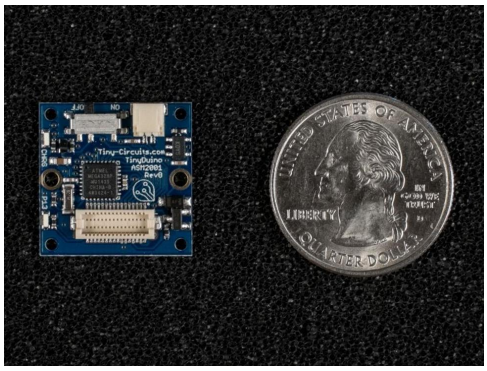


Figure 38: TinyDuino Microprocessor

The TinyDuino is used alongside prebuilt "TinyShields" such as an OLED screen with buttons, a magnetometer and accelerometer to create the diver interface that can display time, temperature and tilt corrected compass heading. The software was written in Arduino IDE and can be found at $< https://github.com/ymrandall/DiverInterface/ >$.

In initial software iterations, C++ in the QT development environment was used to create a diver interface. QT is a coding framework for application development that is made by The QT Company. This framework allows for streamlined coding in C++ in building applications, programs and graphic user interfaces suited for various hardware platforms, including iOS, Android and windows (QT 2017).

C++ was chosen initially as the programming language for several reasons. The primary reason is that it was deemed as the industry standard. Using C++ gives more flexibility in the future to add new technologies. For example the use of C++ allows us to interact with the SeaTrac Modem already made software and use the well developed QT development suite.

8.8 Underwater Housing

In order to account for uniquely sized and shaped hardware, a custom underwater housing was designed in SolidWorks as seen in Figure 39. The housing measures 28mm in width, 47 mm in length and 26 mm in height and is fitted with a 1.6mm thick acrylic sheet. The housing utilizes two spring operated buttons and o-rings salvaged from a GoPro Housing in order to access the two small buttons located on either side of the OLED screen inside the housing. In the model a small hole can be seen which is used to pot out the tether. The loops on the top and bottom of the case are used to string Velcro through to be attached to the divers wrist.

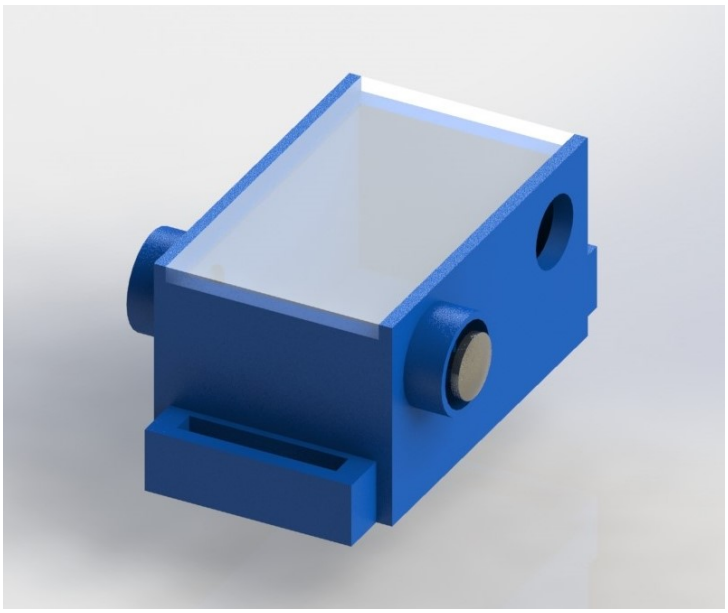


Figure 39: Underwater Housing Model

The housing is printed in ABS plastic and then waterproofed using an acetone based spray paint. The acrylic sheet is sealed on using slow-cure epoxy.

8.9 Moving Forward

Some considerations must be made for future improvements in the diver watch system. The diver watch should have pre-programmed messages for simple communication between divers. To allow for access in order to modify the dive watch components, the underwater housing should not permanently seal the contents within.

9 Conclusion

Current underwater archaeological dredging standards call for the use of a costly topside support research vessel with a hull-mounted pump that requires a licensed boat captain and deckhands. The objective of the DigSki is to provide an affordable research vessel and dredging solution, that can be operated solely by an underwater archaeology team.
A dredge system was designed to be integrated with a PWC. In field testing was done with a designed PWC-operated dredge system at the University of Rhode Island Sailing Center. The designed interface

connecting a PWC and a dredge was successful and a dredge was able to be operated using the PWC proving the concept. Though dredging with a PWC was successful, because of the redirection closed system issue, inconsistent volume flow rates were observed throughout the system. Additionally, the PWC's impeller's output volume flow rate was more than 230 GPM, which far exceeded expectations. In order to accommodate this flow, the jet pump would need to be resized accordingly. A PWC-operated closed dredge system is viable with the suggested modifications.

Diver safety is of the utmost importance, therefore at minimum a diver-operated emergency PWC shut-off was required. This also ensures that the diver will never have to surface near a running engine. This emergency shut-off unit is tethered to the PWC, and will remain close to the divers at all times. Additionally, within this unit, control capabilities were increased by including engine speed control. At a subsurface level, a wearable diver interface was created to work alongside an acoustic modem to provide diver-to-diver ranging and communication. This provides an affordable, small solution to a typically costly system.

10 Nomenclature

10.1 Variables

A	Area (m^2)
g	Gravity ($\frac{m}{s^2}$)
CR	Cavitation Resistance
D	Diameter (m)
K	Friction Loss Coefficient
M	liquid over liquid Flow Ratio ($\frac{Q_2}{Q_1}$)
N	Pressure Ratio, Jet Pump
P	Pressure: static, total ($kPa\ abs.$)
P_v	Vapor Pressure ($kPa\ abs.$)
P_{atm}	Atmospheric Pressure ($kPa\ abs.$)
Q	Volumetric Rate ($\frac{m^3}{s}$)
S	Density Ratio ($\frac{\rho_2}{\rho_1}$)
V	Velocity ($\frac{m}{sec}$)
Z	Jet Dynamic Pressure (m^2)
a	Diffuser Area Ratio ($\frac{A_t}{A_d}$)
b	Jet Pump Area Ratio ($\frac{A_n}{A_t}$)
c	Area Ratio of Throat and Nozzle ($\frac{1-b}{b}$)
m	Mass Flow Rate ($\frac{kg}{s}$)
s	Seconds (s)
sp	Nozzle-to-Throat Spacing (m)
$\frac{sp}{D_{th}}$	Spacing, Throat Diameters
η	Efficiency

10.2 Subscripts

1	Liquid Primary Flow
2	Liquid Secondary Flow
mep	Maximum Efficiency Point
op	Operating Point

L	Limit for Cavitating Flow
3	Combined Fluids 1, 2
i, s, n	Locations
$0, t, d$	Locations
f	Friction Loss
n	Nozzle
en	Throat Entry
th	Mixing Throat
di	Diameter
td	Throat and Diffuser

References

Barratt, J.R. and R.P. Sharphouse. *SeaTrac Product Catalogue*. 2017. URL: https://www.blueprintsubsea.com/seatrac/products.php.

Boyce, Joseph I. et al. *Marine Magnetic Survey of a Submerged Roman Harbour, Caesarea Maritima, Israel*. 2004.

Buxton, Bridget. *King Herod's Harbor*. Oct. 2017.

Chardon, Patricia and Miguel Canals. "Jetski-Based Bathymetric Surveying in Rincon, Puerto Rico". PhD thesis. University of Puerto Rico at Mayaguez, 2012.

Honeywell. *COMPASS HEADING USING MAGNETOMETERS*. URL: https://aerocontent.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/Magnetic_Literature_Application_notes-documents/AN203_Compass_Heading_Using_Magnetometers.pdf.

Karassik, Igor J et al. *Pump Handbook*. 2001.

MacMahan, Jamie. "Hydrographic Surveying from Personal Watercraft". PhD thesis. University of Florida, Feb. 2001, pp. 12–24.

Navigator: Diver Held Sonar Imaging and Navigation System. URL: www.sharkmarine.com/products/diver-held-systems/navigator/.

Neasham, Jeff. *Nanomodems*. 2017. URL: https://udrc.eng.ed.ac.uk/sites/udrc.eng.ed.ac.uk/files/attachments/Towards%20large%20scale%20underwater%20communication%20networks.pdf.

QT. *Cross-platform software development for embedded and desktop*. 2017. URL: https://www.qt.io/.

11 Appendix

```
% Script Description: LJL_Pump_Design.m uses the liquid-jet-liquid model
% from the Pump Handbook to calculate a pump's dimensions and required
% volumetric flow rate and fluid pressure inputs to produce desired suction
% volumetric flow rate.
%
% Assumptions: steady, turbulant, flow.
%
% Units: SI
%
% Coordinate: Origin is at jetski outlet / dredge inlet.
% z is positive below origin.
%% Settting up MATLAB environment
clear
clc
close all
format compact
%% Initial Parameters
% Gravitational Acceleration (m/s^2)
g = 9.81;
% Desired Q2
Q2 = 46.83;              % (gpm)
Q2 = Q2 * 0.0000631;     % (m^3/s)
% Power Fluid Density (kg/m^3)
rho_pf = 1025;
% Wet Sand Density (kg/m^3)
rho_s = 1922;   % 1922
% Suction Fluid Density (kg/m^3)
rho_sf = (0.75*rho_pf)+(0.25*rho_s);
% Design Cavitation Coefficient
sigma = 1.35;
% Diffuser Angle (degrees)
diff_ang = 5;
% Flow Ratio (Q2/Q1)
M = 0:0.00025:2.8;
% (Nozzle / Throat) Area Ratio
b = 0.00125:0.00125:0.9;
% Area Ratio
c = (1-b)./b;
% Area Ratio a^2 = (Ath/Adi)^2 (assuming Ddi = 2 Dth)
% a_2 = ((1^2)/(2^2))^2;
a_2 = 0.0406;
% Nozzle Friction-Loss Coefficient
Kn = 0.05;
% Nozzle Entry Friction-Loss Coefficient
Ken = 0;
% Kth+Kdi, Throat + Diffuser Friction-Loss Coefficients
Ktd = 0.2;
% Opertional Depth or Location of Jet Pump (m)
h_op = 0;
h_op = 3.43; % to get Pd of 5 psi
h_op = 6;
% Suction Lift Height (m)
hs = 0.91;
hs = 1.52;
hs = 1;
% Atmospheric Pressure (Pa)
Pa = 101325;
% Hydrostatic Pressure (Pa)
Ph = rho_pf*g*h_op;
% Suction Lift Pressure (Pa)
Pl = rho_pf*g*hs;
% Absolute Suction Pressure @ s (Pa)
Ps = Pa+Ph-Pl;
% Absolute Diffuser Pressure @ d (Pa)
Pd = Pa+Ph;
```

```
% Primary or Secondary Fluid (whichever is higher) Vapor Pressure (Pa)
Pv = 3490;
% Initial Pressue @ i (Pa)
%Pi_a = 206843;
%% Computation
% Density Ratio
S = rho_sf / rho_pf;
% Preallocating Memory
N = zeros(length(b),length(M));
n = zeros(length(b),length(M));
n_max_index = zeros(1,length(b));
M_me = zeros(1,length(b));
M_op = zeros(1,length(b));
M_op_index = zeros(1,length(b));
N_op = zeros(1,length(b));
n_op = zeros(1,length(b));
Pi = zeros(1,length(b));
M_L = zeros(1,length(b));
Z = zeros(1,length(b));
CR = zeros(1,length(b));
Q1 = zeros(1,length(b));
An = zeros(1,length(b));
Dn = zeros(1,length(b));
Ath = zeros(1,length(b));
Dth = zeros(1,length(b));
sp = zeros(1,length(b));
Lth = zeros(1,length(b));
Ddi = zeros(1,length(b));
Ldi = zeros(1,length(b));
for i = 1:length(b)
    for j = 1:length(M)
        % Term 1 of N Equation
        T1 = 2*b(i);
        % Term 2 of N Equation
        T2 = (2*S*M(j).^2*b(i).^2)./(1-b(i));
        % Term 3 of N Equation
        T3 = - b(i).^2*(1+Ktd+a_2)*(1+M(j))*(1+S*M(j));
        % Term 4 of N Equation
        T4 = - ((S*M(j).^2)./c(i).^2)*(1+Ken);
        % Theoritical Pressure Characteristic
        N(i,j) = (T1+T2+T3+T4)/(1+Kn-(T1+T2+T3+T4));
        % Pump Efficiency
        n(i,j) = M(j)*N(i,j);
    end
    n_max_index(i) = find(n(i,:) == max(n(i,:))); % max n per range of b's
    % Max Efficiency Flow Ratio (Q2/Q1)
    M_me(i) = M(n_max_index(i));                % corresponding M
    % Operable Flow Ratio (Q2/Q1)
    M_op(i) = (1)*M_me(i);
    [val,M_op_index(i)] = min(abs(M-M_op(i)));  % finds M point closest to theortical M_op
    M_op(i) = M(M_op_index(i));
    N_op(i) = N(i,M_op_index(i));
    n_op(i) = n(i,M_op_index(i));
    % Initial Pressure @ i (Pa)
    Pi(i) = ((Pd - Ps)/N_op(i))+Pd;
    % Dynamic Pressure @ nozzle (Pa)
    Z(i) = (Pi(i) - Ps)/(1+Kn);
    % Cavitation Limit Flow Ratio
    M_L(i) = c(i)*sqrt((Ps-Pv)/(sigma*Z(i)));
    % Cavitation Resistance (%)
    CR(i) = ((M_L(i)-M_op(i))/M_op(i))*100;
    % Q1 (m^3/s)
    Q1(i) = Q2/M_op(i);
    % Nozzle Area (m^2)
    An(i) = Q1(i)*sqrt(rho_pf/(2*Z(i)));
    % Nozzle Diameter (m)
    Dn(i) = sqrt((4/pi)*An(i));
    % Throat Area (m^2)
    Ath(i) = An(i)/b(i);
```

```
    % Throat Diameter (m)
    Dth(i) = sqrt((4/pi)*Ath(i));
    % Nozzel to Throat Spacing (m)
    sp(i) = Dth(i);
    % Throat Length (m)
    Lth(i) = 6*Dth(i);
    % Diffuser Diameter (m)
    Ddi(i) = 2*Dth(i);
    % Diffuser Length (m)
    Ldi(i) = ((Ddi(i)-Dth(i))/2)/tand(diff_ang);
end
%%
fig = figure;
set(fig,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
yyaxis left
plot(b,M_op,'-','LineWidth',1.5,'color',[0.2734375 0.5078125 0.703125])
hold on
p=plot(b(1:20:end),M_op(1:20:end),'*','LineWidth',0.5,'color',[0.2734375 0.5078125 0.703125]);
set(get(get(p,'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off'); % Exclude line from legend
hold on
plot(b,N_op,'-','LineWidth',1.5,'color',[1.00 0.83984375 0])
hold on
p=plot(b(1:20:end),N_op(1:20:end),'^','LineWidth',0.5,'color',[1.00 0.83984375 0]);
set(get(get(p,'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off'); % Exclude line from legend
ylabel('Operable Flow Ratio M_o_p & Pressure Ratio N_o_p','FontSize', 18)
axis([0 0.6 0 3])
yyaxis right
plot(b,n_op,'-','LineWidth',1.5,'color',[1.00 0.078125 0.57421875])
hold on
p=plot(b(1:20:end),n_op(1:20:end),'o','LineWidth',0.5,'color',[1.00 0.078125 0.57421875]);
set(get(get(p,'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off'); % Exclude line from legend
hold on
l=line([0.0855 0.0855],[0 0.3]);
l.LineStyle = '--';
l.LineWidth = 1;
set(get(get(l,'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off'); % Exclude line from legend
ylabel('Operable Efficiency \eta_{op}','FontSize', 18)
xlabel('(Nozzle / Throat) Area Ratio b','FontSize', 18)
leg=legend('M_o_p','N_o_p','\eta_{op}');
leg.FontSize = 14;
xticks(0:0.05:0.6)
grid on
grid minor
%%
figure
xlabel('Flow Ratio M')
yyaxis left
plot(M,N(200,:),'-')
ylabel('Pressure Ratio N')
yyaxis right
plot(M,n(200,:),'-')
ylabel('Efficiency %')
grid
title(['b=' num2str(b(200))])
legend('N','\eta %')
grid on
grid minor
%%
figure
Pi = Pi.*0.000145038;
Pi = Pi - 14.7;
plot(b(80:end),Pi(80:end),'LineWidth',1)
ylabel('Initial Pressure P_i','FontSize', 18)
xlabel('(Nozzle / Throat) Area Ratio b','FontSize', 18)
grid on
```

```
grid minor
%%
% figure
% xlabel('Flow Ratio M')
% yyaxis left
% plot(M,N(300,:),'-')
% ylabel('Pressure Ratio N')
% yyaxis right
% plot(M,n(300,:),'-')
% ylabel('Efficiency %')
% grid
% title(['b=' num2str(b(300))])
% legend('N','\eta %')
%% Testing range of Pi and Q1
%Pi_t = ((Pd - Ps)/N)+Pd;
% Pi = Pd:Pd/10:Pd*10;
% N = (Pd-Ps)./(Pi-Pd);
% Pa = Pa.*0.000145038;
% Pd = Pd.*0.000145038;
% Pi = Pi.*0.000145038;
% Q1 = Q2/10:Q2/10:10*Q2;
% M = Q2./Q1;
% figure
% plot(Pi,N)
% hold on
% plot(Pi(1:1:10),N(1:1:10),'o')
% grid on
% xticks(0:25:300)
%%
% N = linspace(0.25,1,20);
% Pi = ((Pd-Ps)./N)+Pd;
% Pi = Pi.*0.000145038;
% Pd = Pd.*0.000145038;
% Ps = Ps.*0.000145038;
% Q2 = Q2 / 0.0000631;
% Q1 = Q1(200) / 0.0000631;
%%
% Pi = [Pd+(Pd/50) Pd*5 Pd*10];
% N = (Pd-Ps)./(Pi-Pd);
% Pi = Pi.*0.000145038;
%
% figure
% Pi = Pi.*0.000145038;
% plot(b,Pi,'-','LineWidth',1.5)
% yticks(0:10:100)
% axis([0 0.6 20 100])
% ylabel('Pressure @ Inlet \P_{i}','FontSize', 18)
% xlabel('(Nozzle / Throat) Area Ratio b','FontSize', 18)
%% Model Compared to Commercial Spec Sheet
% Commercial
Q2_com = [5.85 8.1 9.5 10 12 12 12 12]*4;
Q1_com = [3.55 5 6.1 7.1 7.9 8.7 10 11]*4;
M_com = Q2_com./Q1_com;
% M_com = fliplr(M_com);
Pi_com = [10 20 30 40 50 60 80 100];
% Pi_com = fliplr(Pi_com);
% Model
N_mod = N(200,:);
Pi_mod = ((Pd - Ps)./N_mod)+Pd;
Pi_mod = Pi_mod.*0.000145038;
Pi_mod = Pi_mod - 14.7;
% Ploting
figure
% plot(M_com,Pi_com,'-^b','LineWidth',1)
% hold on
% plot(M(1:50:end),Pi_mod(1:50:end),'-or','LineWidth',1)
% axis([1.09 1.65 0 100])
% xlabel('Flow Rate Ratio M (Q2/Q1)','FontSize', 18)
% ylabel('Input Pressure P_i (Psi gauge)','FontSize', 18)
```

```
% leg = legend('Commercial Specifications','Model');
% leg.FontSize = 18;
plot(Pi_com,Q2_com)
hold on
plot(Pi_com,Q1_com)
legend('Q2','Q1')
grid on
grid minor
%% Model Compared to Commercial Spec Sheet
% % Commercial
% Q2_com = [1.5 3.2 5 7 9 11 11]*4;
% Q1_com = [5.2 6.3 7.2 8 8.7 10 11]*4;
% M_com = Q2_com./Q1_com;
% M_com = fliplr(M_com);
% Pi_com = [20 30 40 50 60 80 100];
% Pi_com = fliplr(Pi_com);
% % Model
% N_mod = N(200,:);
% Pi_mod = ((Pd - Ps)./N_mod)+Pd;
% Pi_mod = Pi_mod.*0.000145038;
% Pi_mod = Pi_mod - 14.7;
% % Ploting
% figure
% plot(M_com,Pi_com,'-^b','LineWidth',1)
% hold on
% plot(M(1:50:end),Pi_mod(1:50:end),'-or','LineWidth',1)
% axis([0.28 1.2 0 100])
% xlabel('Flow Rate Ratio M (Q2/Q1)','FontSize', 18)
% ylabel('Input Pressure P_i (Psi gauge)','FontSize', 18)
% leg = legend('Commercial Specifications','Model','location','northwest');
% leg.FontSize = 18;
% grid on
% grid minor
```

Listing 1: Source code for Jet Pump design.

```
% Script Description: Dredge_Pressure_Drop.m calculates pressure drop of a
% dedge system caused by friction loss, geometric loss, height change, and
% velocity change for all segments in the system using the energy equation.
%
% A segment is defined as a dredge component that has constant diameter
% and material type.
%
% Assumptions: steady, turbulant, flow.
%
% Units: SI
%
% Coordinate: Origin is at jetski outlet / dredge inlet.
% z is positive below origin.
%% Settting up MATLAB environment
clear
clc
close all
format compact
%% Constants
g = 9.81;                           % gravitationl acceleration (m/s^2)
%% Inputs
rho = 1025;                         % seawater density (kg/m^3)
Q = 26.66*1;                        % fluid volume flow rate (gpm)
Q = Q * 0.0000631;                  % fluid volume flow rate (m^3/s)
zi = 0;                             % height of jetski outlet / dredge inlet (m)
zf = 0;                             % height of eductor (m)
%% Importing Segment Properties from Excel
filename = uigetfile('*.xlsx');
data = xlsread(filename);
data = data(:,1);
%% Extracting Data
d = data(3:6:end);                  % pipe diameter (in)
```

```
33  L = data(4:6:end);                    % pipe length (in)
34  f = data(5:6:end);                    % Darcy friction factor
35  K = data(6:6:end);                    % minor loss coefficient
36  %% Converting Data from (in) to (m)
37  d = d*0.0254;                         % pipe diameter (m)
38  L = L*0.0254;                         % pipe length (m)
39  %% Computing Pressure Drop
40  A = (pi*d.^2)/4;                      % pipe area (m^2)
41  v = Q./A;                             % fluid velocity (m/s)
42  dp_v = zeros(length(v),1);
43  for i = 1:length(v)
44      if i == 1
45          dp_v(i) = 0;
46      else
47      dp_v(i) = 0.5*rho*(v(i)^2-v(i-1)^2);   % pressure drop due to v (Pa)
48      end
49  end
50  dp_v_T = sum(dp_v);
51  dp_z = rho*g*(zf-zi);                 % pressure drop due to z (Pa)
52  dp_l = (rho/2)*v.^2.*((f.*L./d)+K); % pressure drop due to losses (Pa)
53  dp_l_T = sum(dp_l);                   % total pressure drop due to loss (Pa)
54  dp_T = dp_z + dp_v_T + dp_l_T;          % total pressure drop (Pa)
55  dp_T_psi = dp_T * 0.000145038;        % total pressure drop (Psi)
56  %%
57  disp('Total Pressure Drop (psi):')
58  disp(dp_T_psi)
```

Listing 2: Source code for Pressure Drop Analysis.

Miller Plastics Eductor Capacity Chart

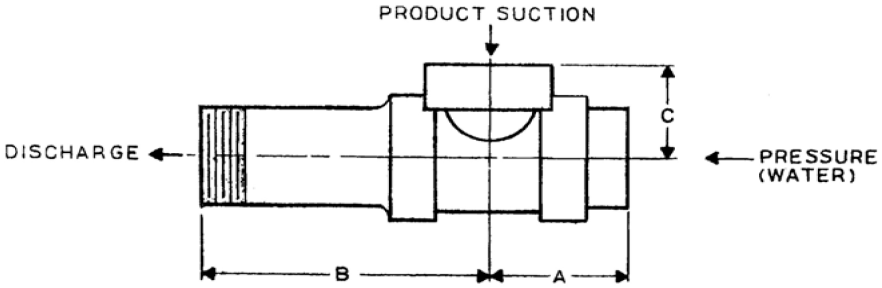
* SUCTION CAPACITY OF STANDARD 1" EDUCTOR - GPM

SUCTION LIFT FT. OF WATER	DISCHARGE PRESSURE	WATER CONSUMPTION GPM	OPERATING WATER PRESSURE - PSI GAUGE							
			10	20	30	40	50	60	80	100
0	0	SUCTION	5.85	8.1	9.5	10.0	12.0	12.0	12.0	12.0
		OPERATING	3.55	5.0	6.1	7.1	7.9	8.7	10.0	11.0
	5	SUCTION		1.4	4.1	6.0	8.0	10.0	11.0	12.0
		OPERATING		4.9	6.1	7.0	7.9	8.6	10.0	11.0
	10	SUCTION			0.28	2.3	4.8	6.4	8.8	11.0
		OPERATING			5.9	6.8	7.8	8.5	9.8	11.0
	15	SUCTION					1.2	3.4	5.9	8.6
		OPERATING					7.7	8.4	9.8	11.0
	20	SUCTION						0.3	3.5	5.9
		OPERATING						8.2	9.7	11.0
	25	SUCTION							0.83	3.9
		OPERATING							9.6	11.0
	30	SUCTION								1.7
		OPERATING								11.0
5	0	SUCTION	4.4	6.8	8.6	9.6	11.0	11.0	12.0	12.0
		OPERATING	3.9	5.3	6.4	7.3	8.1	8.8	10.0	11.0
	5	SUCTION		1.5	3.2	5.0	7.0	9.0	11.0	11.0
		OPERATING		5.2	6.3	7.2	8.0	8.7	10.0	11.0
	10	SUCTION				1.9	3.6	5.6	8.6	10.0
		OPERATING				7.1	7.9	8.6	10.0	11.0
	15	SUCTION					1.1	2.6	5.8	8.3
		OPERATING					7.8	8.6	9.9	11.0
	20	SUCTION							3.3	5.6
		OPERATING							9.8	11.0
	25	SUCTION							0.47	3.6
		OPERATING							9.8	11.0
	30	SUCTION								1.5
		OPERATING								11.0
10	0	SUCTION	2.0	4.6	6.7	8.3	9.0	10.0	10.0	10.0
		OPERATING	4.2	5.5	6.6	7.4	8.2	9.0	10.0	11.0
	5	SUCTION			2.0	4.3	5.9	7.7	9.9	10.0
		OPERATING			6.5	7.4	8.2	8.9	10.0	11.0
	10	SUCTION				1.1	3.0	4.5	8.1	9.6
		OPERATING				7.3	8.1	8.8	10.0	11.0
	15	SUCTION					1.1	2.1	5.6	7.3
		OPERATING					8.0	8.7	10.0	11.0
	20	SUCTION							2.8	5.3
		OPERATING							9.9	11.0
	25	SUCTION								2.8
		OPERATING								11.0
	30	SUCTION								1.1
		OPERATING								11.0
15	0	SUCTION		3.3	5.3	7.9	8.4	8.9	8.9	9.1
		OPERATING		5.7	6.8	7.6	8.4	9.1	10.0	12.0
	5	SUCTION				4.0	4.9	7.3	8.6	9.1
		OPERATING				7.6	8.3	9.0	10.0	11.0
	10	SUCTION					2.4	4.0	6.4	8.6
		OPERATING					8.2	9.0	10.0	11.0
	15	SUCTION							4.2	6.8
		OPERATING							10.0	11.0
	20	SUCTION							2.1	4.5
		OPERATING							10.0	11.0
	25	SUCTION								1.9
		OPERATING								11.0
20	0	SUCTION		2.0	4.0	6.4	7.8	7.8	7.8	7.8
		OPERATING		6.0	7.0	7.8	8.6	9.3	11.0	12.0
	5	SUCTION				2.8	3.9	6.3	7.8	7.8
		OPERATING				7.7	8.5	9.2	10.0	12.0
	10	SUCTION					1.2	3.1	5.7	7.1
		OPERATING					8.3	9.1	10.0	12.0
	15	SUCTION							3.6	5.4
		OPERATING							10.0	11.0
	20	SUCTION							1.4	3.8
		OPERATING							10.0	11.0
	25	SUCTION								1.5
		OPERATING								11.0

* FOR EDUCTOR SIZES OTHER THAN 1"

MULTIPLY ABOVE CAPACITIES
BY PROPER RATIO FACTOR

EDUCTOR SIZE	1/2"	1"	1-1/2"	2"
RATIO FACTOR	0.36	1	2.89	4



MAXIMUM PRESSURE: 125 PSI AT 90 deg f or 32 deg c

Patent pending

DiverInterface

Introduction

This is the wearable diver interface program designed for TinyScreen. This code allows the user to acoustically range other divers. This device uses several compnenets as detailed below.

Components

This code is built for the tiny circuits, tiny screen package and uses their corresponding libraries. It uses the Bosch BMA250 3-axis accelerometer shield, the Honeywell HMC5883L 3-axis compass shields, and acoustic Nanomodems. A RS232 to TTL converter is needed for acoustic modem interaction. Both the acoustic modems and watch are powered by a 5 volt power source.

Libraries

#include <Wire.h> //https://www.arduino.cc/en/Reference/Wire

#include <SPI.h> //https://www.arduino.cc/en/Reference/SPI

#include <TinyScreen.h> //https://github.com/TinyCircuits/TinyCircuits-TinyScreen_Lib

#include <TimeLib.h> //https://playground.arduino.cc/Code/Time

#include <SoftwareSerial.h> //https://github.com/PaulStoffregen/SoftwareSerial

#include "BMA250.h" //https://tinycircuits.com/products/accelerometer-tinyshield#downloads-anchor

---

Nanomodems

These modems are provided by Jeff Neasham of Newcastle University, School of Electrical and Electronic Egineering. the following is his introduction to the Nanomodems.

Nanomodems are a low cost, miniature acoustic communication and ranging device for underwater vehicles, divers and subsea instruments. Short data messages may be exchanged between units and an efficient “ping” protocol is implemented for range measurement by transponder operation. If multiple units are deployed in known locations, then long baseline positioning (LBL) operation is possible.

Boot Up

Upon boot the dive watch will allow for the storage of several modem addresses. These can be changed to the approiate addresses using the top two (left and right) buttons on the watch and saved accordingly. There is no need to save the adress of the attached modem in the watch. The clock will then be set using the same method as the modem addresses. Upon boot it is necessary to calibrate the compass. This is done by roatating the compass along the three axes (X, Y, Z). The watch is ready for action.

Operation

The watch will allow for acoustic ranging via the minimodems. The upper left button will call this function. It will ask which partner modem to contact. A screen will display succesful function activation. Then the return distance will be displayed. Pressing the indicated button returns the user to the original screen.

Liscense

Contact:

ymrandall@my.uri.edu

jlewis001@my.uri.edu

Special Thanks

Prof. Stephen Licht, University of Rhode Island

Prof. Bridget Buxton, University of Rhode Island

Jeff Neasham, Newcastle University

Listing 3: Diver Interface

```
//--------------------------------------------------------------------------------
//
//  Diver Interface
//  Last Updated 10 May 2018
//
//  For interface with TinyScreen and Nanomodems by Jeff Neasham, Newcastle University
//
//  Written by Yelena Randall and Justin Lewis (URI)
//  Contact: ymrandall@my.uri.edu, jlewis001@my.uri.edu
//
//  MIT License
//
//  Copyright (c) [2018] [Yelena Randall & Justin Lewis - University of Rhode Island]
//  Sponsored by: [Dr. Bridget Buxton]
//
//  Permission is hereby granted, free of charge, to any person obtaining a copy
//  of this software and associated documentation files, to deal
//  in the Software without restriction, including without limitation the rights
//  to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
//  copies of the Software, and to permit persons to whom the Software is
//  furnished to do so, subject to the following conditions:
//
//  The above copyright notice and this permission notice shall be included in all
//  copies or substantial portions of the Software.
//
//  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
//  IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
//  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
//  AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
//  LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
//  OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
//  SOFTWARE.
//
//--------------------------------------------------------------------------------


////////////////////////////////////////////Libraries//////////////////////////////////////////////
#include <Wire.h>
#include <SPI.h>
#include <TinyScreen.h>
#include <TimeLib.h> // Arduino time library - https://playground.arduino.cc/Code/Time
#include <SoftwareSerial.h>
#include "BMA250.h"

////////////////////////////////////////////Variables//////////////////////////////////////////////
// Tiny Screen constants
char brightnessChanged = 0; char brightness = 5; int width=5;

// Analog pins for use with thermistor
const int aPin0 = 0; const int aPin2 = 2;

// Thermistor variables
int Vin, Vin2; float Temp, thermRes, I, Vin_f, Vin_f2, V_therm, l;

// Thermistor constants
const float R2 = 82000;                     // Resistance
const float c = 1500;                       // m/s, sound of speed in water
const float pi = 3.141592654;               // Pi
const float A = 3.412026533*(pow(10,-3));   // Steinhart-hart constants
const float B = -0.9356856656*(pow(10,-4)); // based on temp calibration
const float C = 11.49459621*(pow(10,-7));   // http://www.thinksrs.com/downloads/programs/Therm%20Calc/NTCCalibrator/NTCcalculator.

// Compass variables
int x, y, z, Degrees;

// Compass constants
#define HMC5883_I2CADDR  0x1E               // Define compass
```

```
int x_max=-10000;                          // Starting values for hard iron calibration
int y_max=-10000;                          // These values should be extreme in the
int x_min=10000;                           // opposite direction so it calibrates nicely
int y_min=10000;
int z_max=10000;
int z_min=10000;

// Time 1 sec delay
unsigned long previousMillis = 0;          // To store last time clock was updated
const long interval = 1000;                // 1 Second interval

// Temp 1 sec delay
unsigned long previousMillist = 0;         // To store last time clock was updated

int R;                                     // Range for modems
const int RX = 2;
const int TX = 3;
int count=0;

String pin[] = {"000", "001", "002"};
String pinfin;
int ipin = 0;

// Accelerometer
BMA250 accel;
int ax, ay, az, axh, ayh, pitch, roll;

//////////////////////////////////////////Setup///////////////////////////////////////////
// Indicates version of TinyScreen display
TinyScreen display = TinyScreen(TinyScreenDefault);
SoftwareSerial Serial1(2,3);

void setup() {
  Wire.begin();                            // Initialize I2C
  display.begin();                         // Initialize display
  display.setBrightness(10);               // Set brightness from 0-15
  display.setFlip(1);

  accel.begin(BMA250_range_2g, BMA250_update_time_64ms);//This sets up the BMA250 accelerometer

  HMC5883nit();                            // Initialize compass

  pinMode(aPin0, INPUT);                   // Initialize analog pin 0
  pinMode(aPin2, INPUT);                   // Initialize analog pin 2

  Serial1.begin(9600);

  display.setFont(thinPixel7_10ptFontInfo);// Set text size/font
  display.fontColor(TS_8b_White,TS_8b_Black);// Set text color

  setTime(13,58,00,2,3,2018);              // Set time (hr,min,sec,day,month,year)

  boot();
  pinFinder();
  updateTime();
  compassMessage();

  //Arrow for compass heading
  display.drawLine(2,30,7,20,TS_8b_White);
  display.drawLine(7,45,7,20,TS_8b_White);
  display.drawLine(12,30,7,20,TS_8b_White);

}


//////////////////////////////////////////Main////////////////////////////////////////////
void loop() {
  dispHeading();
  buttonLoop();
```

```
  readTime();
  readTemp();
  delay(100);
}

//////////////////////////////////////////Functions/////////////////////////////////////////
void boot (){
  // to select unit ID to communicate with partner modem
  display.setFont(liberationSansNarrow_16ptFontInfo);// Set text size/font
  display.fontColor(TS_8b_Red, TS_8b_Black);
  display.setCursor(25,22);
  display.print("DigSki");
  delay(2000);
  display.clearScreen();

  while (display.getButtons(TSButtonUpperRight) == 0) { // Right button is used to confirm selection
    display.setFont(thinPixel7_10ptFontInfo);   // Set text size/font
    display.fontColor(TS_8b_Red, TS_8b_Black);
    display.setCursor(10,2);                     // Set position on screen
    display.print("Select unit ID");
    display.fontColor(TS_8b_White, TS_8b_Black); // Font color
    display.setCursor(2,15);
    display.print("<Change | Confirm>");

    if (display.getButtons(TSButtonUpperLeft)) { // Left button is used to cycle through options
      display.fontColor(TS_8b_Red, TS_8b_Black);
      display.setCursor(38,35);
      display.print(pin[ipin]);
      delay(500);
      ipin ++;
    if (ipin>2) ipin = 0;
    }
    }
    pinfin = pin[ipin]; // set pinfin as selected ID
  }

void compassMessage(){
    // Instructions to do a "hard-iron" calibration
    display.clearScreen();
    display.setCursor(2,2);
    display.print("Calibrate compass");
    display.setCursor(2,14);
    display.print("by rotating on all");
    display.setCursor(2,26);
    display.print("three axes of");
    display.setCursor(2,38);
    display.print("device.");
    delay(3000);
    display.clearScreen();
  }

void pinFinder(){
  // query modem to find ping ID, display on screen

  display.fontColor(TS_8b_Blue, TS_8b_Black);
  Serial1.println("$?");

  if (Serial1.available()) {
      String response = Serial1.readString();       // Read serial, response should be
                                                    // #AxxxVyyyy (xxx is node adress,
                                                    // yyyyy is 10-bit battery voltage monitor value)

      String mypin = response.substring(2,4);
      display.setCursor(88,2);
      display.print(mypin);
      }
  }

void readTime(){
```

```
  // Display time

  unsigned long currentMillis = millis();
  display.setFont(liberationSansNarrow_12ptFontInfo);
  display.fontColor(TS_8b_White, TS_8b_Black); // Font color
  display.setCursor(2,2);                      // Set position on screen

  //if (currentMillis - previousMillis >= interval) {
    //previousMillis = currentMillis;           // Save last time clock was updated
    if(hour()<10) display.print(0);            // Print a leading 0 if hour value is less than 0
    display.print(hour());
    display.print(":");
    if(minute()<10) display.print(0);          // Print a leading 0 if minute value is less than 0
    display.print(minute());
    display.print(":");
    if(second()<10) display.print(0);          // Print a leading 0 if seconds value is less than 0
    display.print(second());
    display.print(" ");                        // Empty space after seconds
  //}
}

void updateTime(){
  // Set time for bootup process

  display.clearScreen();
  display.setFont(thinPixel7_10ptFontInfo);    // Set text size/font
  display.fontColor(TS_8b_Red, TS_8b_Black); // Font color
  display.setCursor(20,2);
  display.print("SET TIME");
  display.fontColor(TS_8b_White, TS_8b_Black); // Font color
  display.setCursor(2,15);
  display.print("<Change | Confirm>");
  display.setCursor(15,40);
  display.print("Set Hour:");
  int myhour=hour();

  // Right button to confirm, left to cycle through options

  while(display.getButtons(TSButtonUpperRight) == 0) {
    display.setCursor(75,40);
    if(myhour<10) display.print("0");
    display.print(myhour);
    if(display.getButtons(TSButtonUpperLeft))
      myhour++;
    if(myhour>23) myhour = 0;
    delay(200);
    }

  setTime(myhour, minute(), second(),day(),month(),year());
  delay(500);
  display.clearScreen();
  display.setCursor(10,25);
  display.print("Hour Saved");
  delay(1000);

  display.clearScreen();
  display.fontColor(TS_8b_Red, TS_8b_Black); // Font color
  display.setCursor(20,2);
  display.print("SET TIME");
  display.fontColor(TS_8b_White, TS_8b_Black); // Font color
  display.setCursor(2,15);
  display.print("<Change | Confirm>");
  display.setCursor(15,40);
  display.print("Set Minutes:");
  int myminute=minute();

  while(display.getButtons(TSButtonUpperRight) == 0) {
    display.setCursor(75,40);
    if(myminute<10) display.print("0");
```

```
    display.print(myminute);
    if(display.getButtons(TSButtonUpperLeft))
      myminute++;
    if(myminute>59) myminute = 0;
    delay(200);
    }

  setTime(hour(), myminute, second(), day(), month(), year());
  delay(500);
  display.clearScreen();
  display.setCursor(10,25);
  display.print("Minute Saved");
  delay(1000);

  display.clearScreen();
  display.fontColor(TS_8b_Red, TS_8b_Black); // Font color
  display.setCursor(20,2);
  display.print("SET TIME");
  display.fontColor(TS_8b_White, TS_8b_Black); // Font color
  display.setCursor(2,15);
  display.print("<Change | Confirm>");
  display.setCursor(15,40);
  display.print("Set Seconds:");
  int mysecond=second();
  mysecond = 0;

  while(display.getButtons(TSButtonUpperRight) == 0) {
    display.setCursor(75,40);
    if(mysecond<10) display.print("0");
    display.print(mysecond);
    if(display.getButtons(TSButtonUpperLeft))
      mysecond++;
    if(mysecond>59) mysecond = 0;
    delay(200);
    }

   setTime(hour(), minute(), mysecond, day(), month(), year());
    delay(500);
    display.clearScreen();
    display.setCursor(10,25);
    display.print("Seconds Saved");
    delay(1000);
    display.clearScreen();
  }


void readTemp(){
  // Read temperature from thermistor

  unsigned long currentMillist = millis();

  Vin = analogRead(aPin0);                    // Read V from pin 0
  Vin_f = 5*float(Vin)/1023;                  // Vin forward

  Vin2 = analogRead(aPin2);                   // Read V from pin 1
  Vin_f2 = 5*float(Vin2)/1023;                // Vin2 forward

  thermRes = float(R2*((Vin_f/Vin_f2)-1));    // Resistance across thermistor
  l = log(thermRes);                          // Log calculation for Steinhart-Hart equation
  Temp = (1 / (A + B*l + C*l*l*l))-273.15;    // Temperature in C, Steinhart-Hart Equation

  if (currentMillist - previousMillist >= interval) {
    previousMillist = currentMillist;
    // Display temperature
    display.setFont(thinPixel7_10ptFontInfo);
    display.fontColor(TS_8b_Blue, TS_8b_Black);
    display.setCursor(2,54);
    display.print("Temp: ");
    display.print(Temp);
```

```
    display.print(" *C");
  }
  display.setFont(thinPixel7_10ptFontInfo);// Set text size/font
  display.fontColor(TS_8b_White,TS_8b_Black);// Set text color
}


// Based on TinyCompass demo by Tony Batey
// https://www.hackster.io/tbatey_tiny-circuits/tinycompass-32de65
void HMC5883nit(){
  // Put the HMC5883 3-axis magnetometer into operating mode
  Wire.beginTransmission(HMC5883_I2CADDR);
  Wire.write(0x02);                             // Mode register
  Wire.write(0x00);                             // Continuous measurement mode
  Wire.endTransmission();
}

void dispHeading(){
  // To display compass heading on screen

  display.setFont(thinPixel7_10ptFontInfo);
  display.fontColor(TS_8b_White,TS_8b_Black);
  compassCal();
  display.setCursor(20,22);

  Degrees = Degrees-270;                        // Adjust 270 degrees because the sensor is pointing right
  if (Degrees < 0) Degrees=Degrees+360;
  if(Degrees<30 || Degrees> 330) display.print("N  ");
  if(Degrees >= 30 && Degrees< 60) display.print("NW ");
  if(Degrees >= 60 && Degrees< 120) display.print("W  ");
  if(Degrees >= 120 && Degrees< 150) display.print("SW ");
  if(Degrees >= 150 && Degrees< 210) display.print("S ");
  if(Degrees >= 210 && Degrees< 240) display.print("SE ");
  if(Degrees >= 240 && Degrees< 300) display.print("E  ");
  if(Degrees >= 300 && Degrees< 330) display.print("NE ");

  display.setCursor(20,37);
  Degrees = abs(Degrees - 360);
  display.print(Degrees);                       // Display the heading in degrees
  display.print("  ");
}

void tiltCorrection(){
  // Read accelerometer data

  accel.read();//This function gets new data from the accelerometer
  ax = accel.X;
  ay = accel.Y;
  az = accel.Z;
  pitch = asin(-ax);
  roll = asin(ay/(cos(pitch)));
  }

void compassCal(){
  // calibrate compass

  readMyCompass();                              // Read compass

  if(x > x_max)                                 // Find values of hard iron distortion
    x_max = x;                                  // This will store the max and min values
  if(y >y_max)                                  // of the magnetic field around you
     y_max = y;
  if(y<y_min)
     y_min = y;
  if(x<x_min)
     x_min = x;
  if(z > z_max)
    z_max = z;
  if(z < z_min)
```

```
    z_min = z;

  int xoffset= (x_max+x_min)/2;
  int yoffset= (y_max+y_min)/2;
  int zoffset= (z_max+z_min)/2;

  int x_scale = x-xoffset;                    // Math to compensate for hard
  int y_scale = y-yoffset;                    // iron distortions
  int z_scale = z-zoffset;

  tiltCorrection();

  // incorporate accelerometer data to account for small tilts of the device in pitch and roll
  axh = (x_scale*(1-sq(pitch))) - (y_scale*pitch*roll) - (z_scale*pitch*sqrt(1-sq(pitch)-sq(roll)));
  ayh = (y_scale*sqrt(1-sq(pitch)-sq(roll))) - (z_scale*roll);

  float heading = atan2(axh,ayh);            // Heading in radians

  // Heading between 0 and 6.3 radians
  if(heading < 0)
    heading += 2*PI;

  if(heading>2*PI)
    heading -= 2*PI;

  Degrees = heading * 180/M_PI;               // Conversion to degrees
}

void readMyCompass(){
  // read compass

  Wire.beginTransmission(HMC5883_I2CADDR);
  Wire.write(byte(0x03));                     // Send request to X MSB register
  Wire.endTransmission();
  Wire.requestFrom(HMC5883_I2CADDR, 6);       // Request 6 bytes; 2 bytes per axis

  if(Wire.available() <=6){                   // If 6 bytes available
    x = (int16_t)(Wire.read() << 8 | Wire.read());
    z = (int16_t)(Wire.read() << 8 | Wire.read());
    y = (int16_t)(Wire.read() << 8 | Wire.read());
  }
}

void pingResponse() {
  // Modem ping function

  while (display.getButtons(TSButtonUpperRight) == 0) {

    if (Serial1.available()) {
      String res = Serial1.readString();       // Read serial, response should be
                                               // #RxxxTyyyyy (xxx is unit tag,
                                               // yyyyy used to calculate range)

      String character = res.substring(6);     // Store data into character string after 6

      y = character.toInt();                   // Convert to int for calc

      R = y * c * 6.25 * exp(-5);              // Calculate range in meters
      display.clearScreen();
      display.setCursor(38, 32);               // Display position
      display.println("Range: ");
      display.println(R);
      display.setCursor(14, 10);               // Display position
      display.println("Press to return>");
    }
  }
  display.clearScreen();

  // Redraw arrow for compass heading
```

```
  display.drawLine(2,30,7,20,TS_8b_White);
  display.drawLine(7,45,7,20,TS_8b_White);
  display.drawLine(12,30,7,20,TS_8b_White);
}

void buttonLoop() {
  // Function to send ping command
  if (display.getButtons(TSButtonUpperLeft)) {
    display.clearScreen();
    display.setCursor(0,0);
    display.println("Pinged!");
    Serial1.print("$P000");
    pingResponse();
    delay(500);
  }
}
```

Listing 4: Dredge control panel

```
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>

 /* wiring for LCD screen:
 * LCD R/W pin to ground
 * LCD VSS pin to ground
 * LCD K pin to ground
 * LCD VDD pin to 5V
 * LCD A pin to 5V
 * 10K resistor:
 * ends to +5V and ground
 * wiper to LCD VO pin (pin 3)
*/
//LCD pins
int RS=2;
int Enable=3;
int D4=4;
int D5=5;
int D6=6;
int D7=7;

// Define button pins
int stopbutton=14;
int startbutton=13;
int inpowerbutton=16;
int decpowerbutton=15;

//Led pins
int greenpin=20;
int yellowpin=19;
int redpin=18;

//variables that are used
int sendstop=0;
int sendstart=0;
int sendinpower=0;
int senddecpower=0;

//Delay needed between messages min needed: 1100
int messagedelay=1100;

int dredgepower=0;
String message;
String prntmessage;


// initialize the LCD Screen with pins listed in header
LiquidCrystal lcd(RS, Enable, D4, D5, D6, D7);
```

```
void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  //start communication with topside
  Serial2.begin(9600);
  //setup start and stop button inputs
  pinMode(stopbutton,INPUT);
  pinMode(startbutton,INPUT);
  pinMode(inpowerbutton,INPUT);
  pinMode(decpowerbutton,INPUT);
  pinMode(greenpin,OUTPUT);
  pinMode(yellowpin,OUTPUT);
  pinMode(redpin,OUTPUT);

  //Print labels for data on screen
   lcd.setCursor(0, 0);
   lcd.print("Status:");

   lcd.setCursor(0, 1);
   lcd.print("Power:");

   lcd.setCursor(11, 0);
   lcd.print("Fuel:");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):

  sendstop=digitalRead(stopbutton);
  sendstart=digitalRead(startbutton);
  sendinpower=digitalRead(inpowerbutton);
  senddecpower=digitalRead(decpowerbutton);

//what happens when buttons are pressed
  if (sendstop==HIGH){
    Serial2.println("stop");
    digitalWrite(redpin,HIGH);
    delay(messagedelay);
    digitalWrite(redpin,LOW);}

  if (sendstart==HIGH){
    Serial2.println("start");
    digitalWrite(greenpin,HIGH);
    delay(messagedelay);
    digitalWrite(greenpin,LOW);}

  if (sendinpower==HIGH){
    Serial2.println("inpower");
    digitalWrite(yellowpin,HIGH);
    delay(messagedelay);
    digitalWrite(yellowpin,LOW);}

  if (senddecpower==HIGH){
    Serial2.println("decpower");
    digitalWrite(yellowpin,HIGH);
    delay(messagedelay);
    digitalWrite(yellowpin,LOW);}

  if (Serial2.available()){
    message=Serial2.readString();

    //uncomment to see serial2 message on screen
    //lcd.setCursor(0, 1);
    //lcd.print(message);
    if (message.startsWith("off")){
      lcd.setCursor(7, 0);
      lcd.print("off");
     }
```

```
    else if (message.startsWith("on")){
      lcd.setCursor(7, 0);
      lcd.print("on␣");
      }

    else if (message.startsWith("power")){
      lcd.setCursor(6, 1);
      prntmessage=message.substring(5,(message.length()-2));
      lcd.print("␣␣␣");
      lcd.setCursor(6, 1);
      lcd.print(prntmessage);}

    else if (message.startsWith("fuel")){
      lcd.setCursor(13, 1);
      prntmessage=message.substring(4,(message.length()-2));
      lcd.print("␣␣");
      lcd.setCursor(13, 1);
      lcd.print(prntmessage);}
   }}
```

Listing 5: Dredge control topside

```
// include the library code:
#include <SoftwareSerial.h>

//Adjustable Parrameters:
//************************************************************
//time before start command stops the starter
int starttime=5; //(seconds)

//Delay needed between messages min: 1100
int messagedelay=1100;

//set servo min and max
//location max is 256 as it is 8 bit
int servomax=250;
//location min is 0 but does not read well below 100
int servomin=100;

//Changing steps changges the amount of steps between min and max on the servo
int steps=3;
//************************************************************

//makes steps between max and min for servo
int stepcount=(servomax-servomin)/steps;

// Start and stop relay pins
int stoprelay=19;
int startrelay=20;
//Sensor input pins
int fuelsensor=23;
int enginecheck=22;
//servo pin
int servopin=3;

//Define Variables used
int fuelread=0;
int fuelreadnew=0;
int enginestat=0;
int enginestatnew=0;
int servoposition=servomin;


String message;
char sendmessage[10];
```

```
void setup() {
   //setup start and stop button inputs
  pinMode(fuelsensor,INPUT);
  pinMode(enginecheck,INPUT);
  pinMode(stoprelay,OUTPUT);
  pinMode(startrelay,OUTPUT);
  pinMode(servopin,OUTPUT);
  pinMode(13,OUTPUT);
  //turn off engine when system is turned on
  digitalWrite(stoprelay,HIGH);

  //delay for controller to start and be ready to recieve messages
  delay(2000);

  //start communication with controller
  Serial2.begin(9600,SERIAL_8N1);

  //set location of servo to min value for lowest power
  analogWrite(servopin,servomin);

  //tell controller current state of topside
  sprintf(sendmessage,"power%d",int(100*(servoposition-servomin)/(servomax-servomin)));
  Serial2.println(sendmessage);
  delay(messagedelay);
  Serial2.println("off");
  }

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  // fuelreadnew=analogRead(fuelsensor);
  enginestatnew=digitalRead(enginecheck);
  //send updated data every second

//  if (fuelread!=fuelreadnew){
//    fuelread=fuelreadnew;
//    sprintf(sendmessage,"fuel%d",int(fuelread/2.56));
//    Serial.println(sendmessage);
//    delay(100);
//    }
//  if(enginestat!=enginestatnew){
//    enginestat=enginestatnew;
//    if (enginestat==1){
//      Serial2.println("on");
//      delay(100);}
//    if (enginestat==0){
//      Serial2.println("off");
//      delay(100);}}

///recieving messages from controller
  if (Serial2.available()){
    message=Serial2.readString();

///start message to start PWC engine
    if (message.startsWith("start")){
      digitalWrite(stoprelay,LOW);
      digitalWrite(startrelay,HIGH);
      delay(starttime*1000);
      digitalWrite(startrelay,LOW);
      Serial2.println("on");
      enginestat=1;}

 //stop message to stop PWC engine
    else if (message.startsWith("stop")){
      digitalWrite(startrelay,LOW);
      digitalWrite(stoprelay,HIGH);
      servoposition=servomin;
      analogWrite(servopin,servoposition);
```

```
      sprintf(sendmessage,"power%d",int(100*(servoposition-servomin)/(servomax-servomin)));
      Serial2.println(sendmessage);
      delay(messagedelay);
      Serial2.println("off");
      enginestat=0;}

  //increase power to engine by pulling on throttle cable with servo
    else if (message.startsWith("inpower") && (servoposition<servomax) && (enginestat==1)){
      servoposition+=stepcount;
      analogWrite(servopin,servoposition);
      sprintf(sendmessage,"power%d",int(100*(servoposition-servomin)/(servomax-servomin)));
      Serial2.println(sendmessage);
      delay(messagedelay);}

    //decrease power to engine by pulling on throttle cable with servo
    else if (message.startsWith("decpower") && (servoposition>servomin) && (enginestat==1)){
      servoposition-=stepcount;
      analogWrite(servopin,servoposition);
      sprintf(sendmessage,"power%d",int(100*(servoposition-servomin)/(servomax-servomin)));
      Serial2.println(sendmessage);
      delay(messagedelay);
      }
}}
```

Nanomodem v2.1

User guide

Jeff Neasham
School of Electrical & Electronic Engineering

1. Introduction

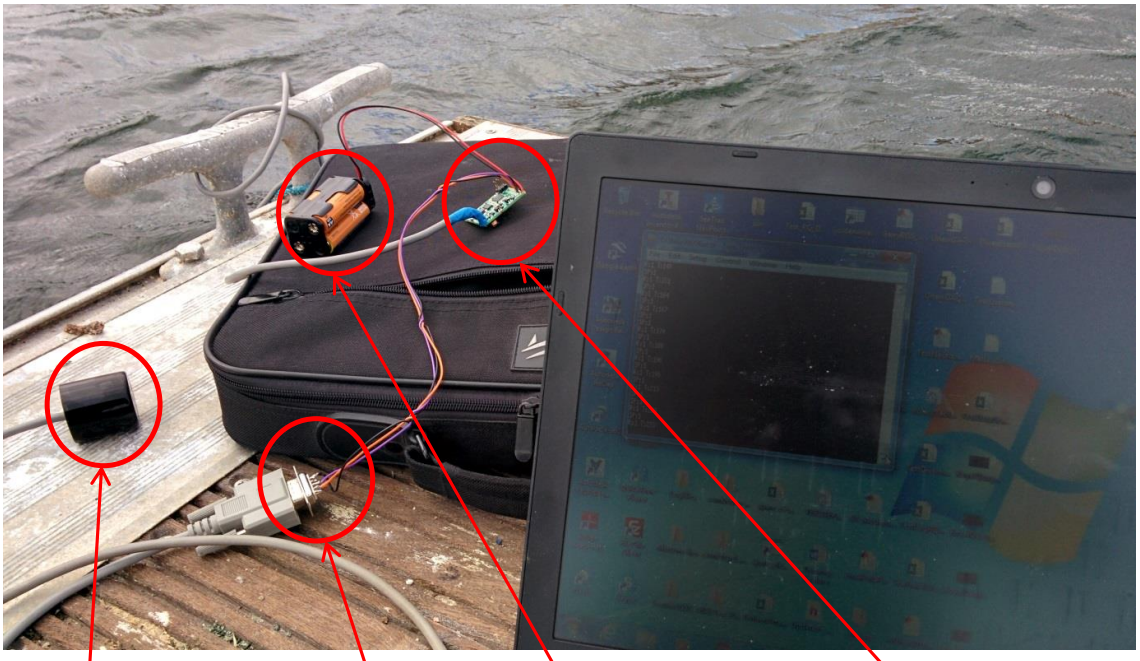
Nanomodems are a low cost, miniature acoustic communication and ranging device for underwater vehicles, divers and subsea instruments. Short data messages may be exchanged between units and an efficient "ping" protocol is implemented for range measurement by transponder operation. If multiple units are deployed in known locations, then long baseline positioning (LBL) operation is possible.

This document describes the operation, electrical interfacing and protocols for these devices.

2. Specification

Supply voltage	3 – 6.5V dc
Supply current (5V supply)	Receiving: max 2mA Transmitting: max 400mA
Acoustic frequency	24-28kHz
Acoustic source level	168 dB re 1uPa @ 1m
Acoustic data rate	40 bits/s, unicast and broadcast data messages up to 7 bytes in length.
Addressing	Up to 256 units (addresses 0-255)
Ranging increment	9.375 cm (c=1500m/s)
Ranging variance	~20 cm
Maximum Range	> 2 km
RS232 interface	9600 Baud, 8-bit, no parity, 1 stop bit, no flow control

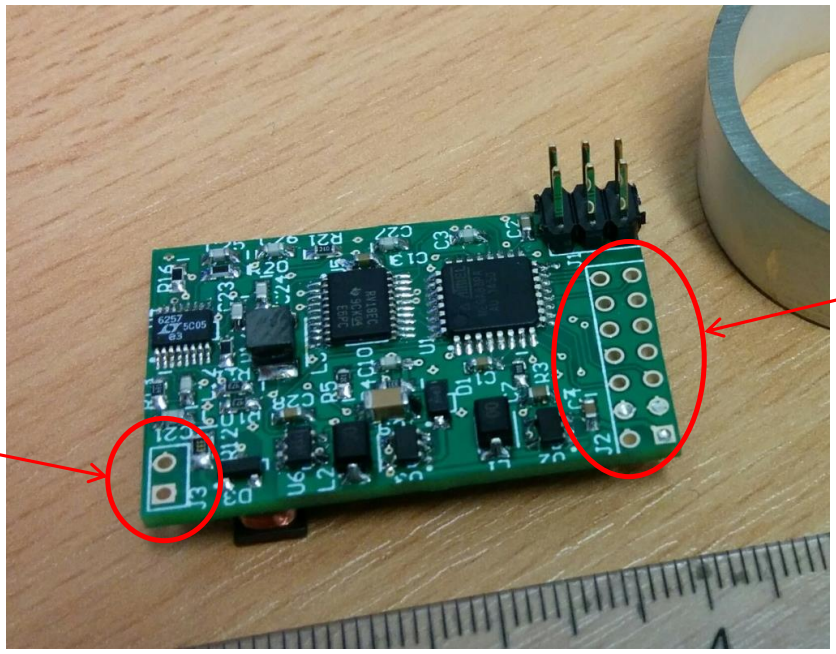
3. Components



Nanomodems may be supplied in 2 different forms:

1. Modems comprising transducer on a 5m cable and a bare PCB for integration with existing housings.
2. A single polyurethane moulded unit incorporating transducer and electronics with 4 way flying lead for power supply and data interface.

Connection details are provided below:



J2 – Main connector		J3 - Transducer	
1	Vbat (+ve supply) *	1	Inner electrode (red wire)
2	Vbat (+ve supply) *	2	Outer electrode (black wire)
3	GND (0V) *		
4	GND (0V) *		
5	Serial TXD (RS232 output)		
6	I/O 1 (not yet implemented)		
7	Serial RXD (RS232 input)		
8	I/O 2 (not yet implemented)		
9	I/O 3 (not yet implemented)		
10	I/O 4 (not yet implemented)		
11	I/O 5 (not yet implemented)		
12	I/O 6 (not yet implemented)		
13	RxS flag 1 (2.5V logic)		
14	RxM flag 2 (2.5V logic)		

The transducer signal on J3 during transmission will reach 200Vp-p so electrical safety precautions must be observed when the device is activated.

* Care must be taken to connect the power supply/battery with the correct polarity as no reverse polarity protection is provided on board.

4. Connection

1. Connect 6V battery pack or power supply (5V recommended) to each modem. (Each modem will start up in receiving mode and draw no more than 2mA of current).
2. Place each transducer in water.
3. Connect unit(s) via RS232 serial cable to PC running a terminal programme (e.g. hyperterminal, Putty etc) and configured with serial port settings (9600, 8, n, 1).

5. Serial Communication protocol

5.1 Format

Serial communication format is (9600, 8, n, 1) with no flow control. All commands issued to the Nanomodem are prefixed with '$'. All responses from the Nanomodem are prefixed with '#' and terminated with <CR><LF>. Unrecognised or invalid commands return 'E' to indicate an error.
Serial commands may be entered as a contiguous string (one byte immediately after the other) or can be typed in manually from a terminal program. If the delay between entered bytes exceeds ~2s the serial handler will time out and return 'E'.

5.2 Node addressing

Each Nanomodem must be allocated a unique 8-bit node address (0-255) which is stored in EEPROM on the device. This is set and queried via a modem command as described in section 5.2.

5.2 Modem commands

Command string	Description	Response string
$Axxx	Set node address to xxx (ascii decimal e.g. 123)	#Axxx – confirms node address has been set to xxx

$?	Query Nanomodem status	#AxxxVyyyy – where xxx is node address and yyyy is 10-bit battery voltage monitor value. To convert to a voltage:
		v = yyyy * 1.1*6/1024

$Pxxx	Ping unit with address xxx	$Pxxx to acknowledge command. #RxxxTyyyyy is then returned if response is received from unit xxx. Range to unit xxx is given by R = yyyyy * c * $6.25e^{-5}$ where c is the sound velocity (assume 1500 m/s if no data is available).

$Vxxx	Query battery voltage on unit xxx	$Vxxx to acknowledge command. #RxxxVyyyy is then returned if a response is received from unit xxx. To convert to a voltage:
		v = yyyy * 1.1*6/1024

$Uxxxnddd…	Unicast data message. Send n bytes (ddd…) to unit xxx. Data bytes (d) can be any printable or non-printable char.	$Uxxxn to confirm n bytes sent to unit xxx. Unit xxx will output #Unddd… when message has been received.
$Bnddd…	Broadcast data message. Send n bytes (ddd…) to all units. Data bytes (d) can be any printable or non-printable char.	$Bn to confirm n bytes have been broadcast. All units will output #Bxxxnddd… (where xxx is the transmitting unit) when message has been received.

5.2 Acoustic packet durations

The following information can be used to calculate the expected transmission times for various acoustic message exchanges. Acoustic propagation delays should be added in calculating the expected duration of bidirectional data exchanges.

Ping message	0.275s
Command message e.g. ($Vxxx)	0.65s
Data message (n bytes)	0.65s + n * 0.2s

5.3 Acoustic receive flags RxS and RxM

When the start of any acoustic packet is detected by a Nanomodem, the RxS flag is raised. The timing of this rising edge coincides precisely with the detection of the packet header waveform and so it may be used for time difference of arrival (TDOA) estimates where multiple Nanomodems are placed in an array. The RxS flag returns to zero at the end of the acoustic packet.

When a Nanomodem receives a broadcast data message or a unicast data message addressed to that unit, the RxM flag is raised for a short period corresponding to the transmissions of the received serial data. This signal may be used, for example, to wake up connected circuitry from a low power state.