Design, Fabrication, and Characterization of a Sand Burrowing Robot

by

Nana Kwame Okwae

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved April 2020 by the
Graduate Supervisory Committee:

Hamidreza Marvi, Chair
Jungliang Tao
Hyunglae Lee

ARIZONA STATE UNIVERSITY

May 2020

ABSTRACT

Unmanned subsurface investigation technologies for the Moon are of special significance for future exploration when considering the renewed interest of the international community for this interplanetary destination. In precision agriculture, farmers demand quasi-real-time sensors and instruments with remote crop and soil detection properties to meet sustainability goals and achieve healthier and higher crop yields. Hence, there is the need for a robot that will be able to travel through the soil and conduct sampling or in-situ analysis of the subsurface materials on earth and in space. This thesis presents the design, fabrication, and characterization of a robot that can travel through the soil. The robot consists of a helical screw design coupled with a fin that acts as an anchor. The fin design is an integral part of the robot, allowing it to travel up and down the medium unaided. Experiments were performed to characterize different designs. It was concluded that the most energy-efficient speed from traveling down the medium is 20 rpm, while 60 rpm was the efficient speed for traveling up the medium. This research provides vital insight into developing subsurface robots enabling us to unearth the valuable knowledge that subsurface environment holds to help the agricultural, construction, and exploration communities.

To Akomaning Ofosu and Christiana Darko

ACKNOWLEDGEMENTS

This work would not have been possible without support from numerous individuals, especially Dr Hamidreza Marvi. Likewise, I would like to thank Hosain Bagheri, Ben Bethke and everyone at Bio-Inspired Robotics Technology and Healthcare Lab for their unique roles in the beginning and throughout the entire journey of this work. Finally, I would like to thank the MasterCard Foundation Scholarship Program for funding my academic endeavors and providing me with opportunities to fulfill my potentials.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

1.1 Motivation and Overview

Robots with self-propelling capabilities through granular media have recently become of interest to the agriculture, construction, and exploration communities -among others- for their low-cost and low environmental impact technologies[7][3][19]. In precision agriculture, farmers demand quasi-real time sensors and instruments with remote crop and soil detection properties to meet sustainability goals and achieve healthier and higher crop yields.[21]. There has also been a considerable amount of attention given to subsurface exploration on both earth and space[29]. Subsurface exploration about the internal material and soil structure of the planet earth would lead to a better understanding of the environment in which we live in.[17].Celestial bodies such as the moon still have a lot of unanswered questions even though missions to space have shown significant progress in space exploration.[6] [23].There is still a need to know about the origin, chemical compositions and internal structures of the moon [17], and other planets as well. Hence, the need for a robot that will be able to travel underground for exploration, sampling, search-and-rescue, underground construction, and geotechnical sensing and monitoring applications.[8]

Nature has evolved excellent biological features that enhance transportation in the subterranean environment that are worth investigating. For example, the Atlantic razor clam, commonly known as the American jackknife, can produce approximately 10 N of force to pull its valves into the soil[26]. This level of force should enable the clam to submerge to approximately 1-2 cm [30]. However, in reality, the razor clams dig to 70 cm. They achieve this feat by contracting its body[9], it agitates and locally fluidizes the soil, reducing the drag and energetic cost of burrowing[20][24]. Mammals such as armadillo are known to be excellent burrows. The armadillo has five clawed toes on their hind feet, and three to five toes with heaving digging claws on their forefeet.[5][14]. That enables them to build burrows in moist soil in which they live and feed. Their claws enable them to dig the ground while the hind feet as an extractor, taking the sand away from the hole. Badgers also have short, full-bodied, and short legs that enable them to dig. Lastly, moles, which are fossorial animals, have cylindrical bodies, reduced hind limbs, and short forelimb with large paws suitable for digging.

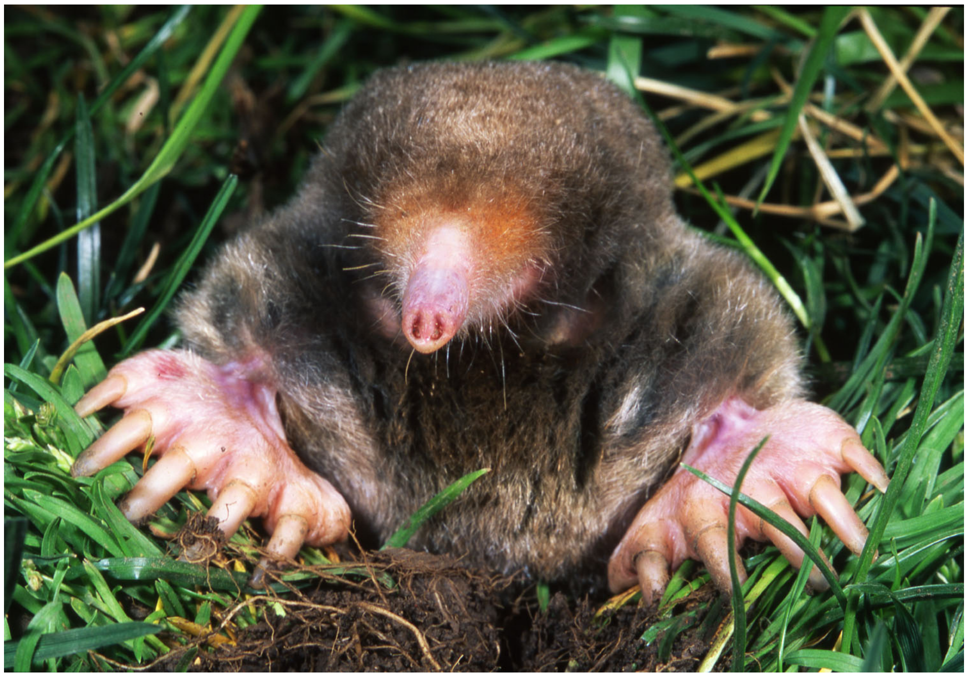


Figure 1.1: Image of a Mole. Courtesy : National Science Foundation [11]

This research aims at designing and fabricating a robot that will be able to successfully burrow the soil to unearth the valuable knowledge that the subsurface environment holds to advance in the agricultural, construction, and exploration communities. This paper also seeks to better existing technologies by optimizing the robot to improve its ability to explore the subsurface environment.

1.2 Overview of Screw Design

The method of transportation to be studied is helical propulsion. Helical patterns are observed in many places in nature, from shell formation to flagellum motion. The helix design has since been incorporated into industries such as oil and gas, seen in rotary drilling. Preliminary research performed on helix design can be applied to a self-propelled robot. The challenges of self-propulsion are producing forward motion and combating media resistance. These challenges beg the question of what design parameters provide the best stability and consume the least power?

Delving into work performed by similar labs offers insight into proof of concept robots, analytical, and experiment optimization that provides the basis for further research into self-borrowing robots. Other helical robots designed for self-propulsion examined consist of a contra-rotor mechanism [18], a bi-modal quadrotor [4], and three helical auger robots [27], [25]. The contra-rotor screw explorer offers a model for cavity expansion yet failed mechanically and did not successfully burrow [18]. The bi-modal self-burying robot was able to dig successfully. The results give reasonable expectations for data to be collected on load, media, and current experiments [4]. Without proper stability, helix propulsion will cause the robot to spin out without any or small forward progress. This emphasizes the need for research into a stability mechanism. Research into the optimization of helical parameters shows the optimal angle of the helix being a consistent 55 degrees. Additional experiments on rotational velocity, loads, and forward velocity yield consistent results and provide expectations for data trends [27],[25]. Learning for these prototypes and accounting for design conditions,a more straightforward design is suggested.

1.3 Overview of Fin Design

The proposed robot consists of a single rotating screw with fins to act as anchors. A single screw simplifies the mechanical design eliminating places for failure. A conical and cylindrical shape contains the electrical components which shield them from the granular media. As rotary drilling anchors to the ground, the idea is to attach fins to the body to anchor to the media preventing full robot rotation, so screw rotation propels the robot through the media, creating a burrowing effect. Research on various species with superior locomotion adaptations gives potential insight into more efficient robots that are worth investigating. The scalloped flipper shape of the humpback whale was noted to decrease drag by 32% compared to that of a smooth shape meaning less energy consumption. The angle of the flipper during maneuvers influences drag was discovered to control yaw [15]. The geometrical structure of the mole rats fore claws toe was found to decrease soil cutting resistance by 12.80% [10]. The excellent movement abilities of these two species show the geometrical parameters of the fins can be optimized to decrease resistance and as a result, decrease power. There are many features to consider when designing the fins. The geometrical shape, how many, arrangement around the screw, length, surface area, angle and media to be borrowed since it can impact the efficiency of the robot and its ability to self burrow.

From the evolved biological systems, it is believed that the concave-convex shape of the fin will decrease resistance from the media compared to flat, smooth geometries. It is anticipated that this bio-inspired geometry will be the optimal design. Applying resistive force theory to the fins, no fins would be optimal because there would be no surface to create drag, so less power would be spent overcoming the drag; however, the robot would not burrow without them. Therefore, we anticipate fewer fins and smaller dimensions being better, but the fins must also have enough support that the load does not break them off.

1.4 Potential Impact

Once optimized, this research will significantly improve the sustainability for industries requiring digging, such as environmental studies, agriculture, and construction. These industries analyze the soil to increase the performance and sustainability of their practices. Current technology requires costly excavation that is environmentally damaging and challenging to maintain. Furthermore, the performance of applied technology decreases quickly due to the harsh environment so that wireless system would be ideal. A low power, self-burrowing robot would significantly affect the sustainability of such industries and demonstrates the necessity to optimize self-burrowing, resistance reducing technology. Helping these industries will in turn positively affect society with cheaper, safer, and environmentally friendly soil analysis, leading to more accurate natural disaster and infrastructure readings among others.

Chapter 2

DESIGN AND FABRICATION OF THE SAND BURROWING ROBOT

2.1 Screw Design

Archimedes screw mechanism is an ancient design that was used for lifting water for irrigation and drainage purposes in Ancient Egypt.[22].In recent times, it has seen a significant revival in modern engineering, by reversing it for use as a turbine instead of a pump.[13].

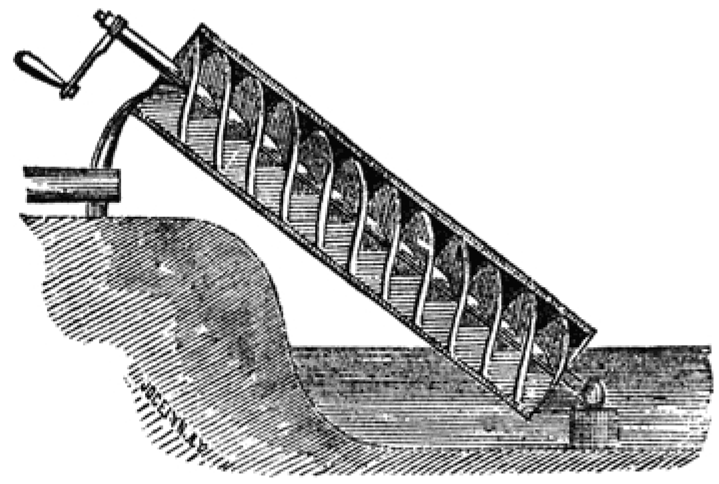


Figure 2.1: Archimedes Screw[2]

There are two types of Archimedean screw mechanisms that were used in the design of the screw for the robot, namely: a logarithmic and a cylindrical helix.

2.2 Geometric Modelling of Screw Mechanism

The mathematical models of the helices must first be defined; these models are based on worked done by Nagoka.[17][18]. In the equations below η denotes the constant inclination angle of screw flight at the center position **P** on the screw. The screw length is represented by L, the maximum inner radius of the screws is r_0 and the maximum screw radius is R_0. The logarithmic and cylindrical screw models are expressed as a function of the winding screw angle θ against a cone and a cylinder. l denotes the height from the apex of the screw, r is defined as the distance from l axis and θ is set to be zero at the highest position of the screws at a winding angle θ,r_c, r_s , and r_{sc} defined as the inner screw radius, the outer screw radius, and the distance from l axis to **P**, respectively. The mathematical models of the helices can be defined as follows:

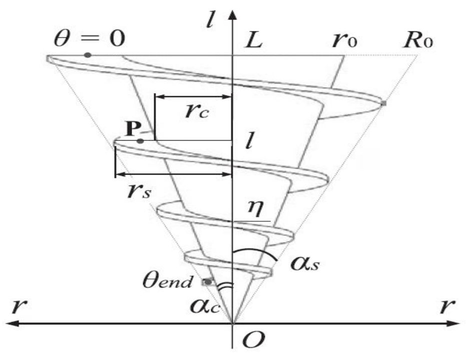
$$\mathbf{R} = \begin{cases} R_L \mathbf{exp(a}\ \theta\mathbf{)} & : \ \textbf{Logarithmic} \ \ \textbf{Helix} \\ R_C & : \textbf{Cylindrical} \ \ \textbf{Helix} \end{cases} \tag{2.1}$$

and also,

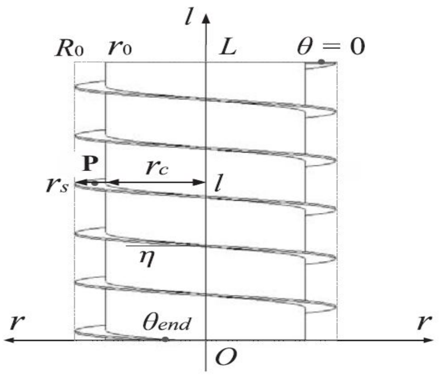
$$\begin{cases} \mathbf{R} & = [r_c(\theta)\ r_s(\theta)\ r_{sc}(\theta)\ l(\theta)] \\ \mathbf{R}_L & = [r_0 R_0 R L]^T \\ \mathbf{R}_C & = [r_0 R_0 R L - \frac{p\theta}{2\pi}]^T \\ R & = [\frac{r_0}{2} + \frac{R_0}{2}] \end{cases} \tag{2.2}$$

where p is the screw pitch and the function parameter a is negative (i.e. , a < 0) The slope parameter a is also defined as:

$$a = \frac{-R\ tan\ \eta}{\sqrt{(L + R\ tan\ \eta)(L - R\ tan\ \eta)}} \tag{2.3}$$



(a) Logarithmic Helix



(b) Cylindrical Helix

Figure 2.2: Geometric Models of Screw Helices[18]

The screw pitch p of a cylindrical helix is constant, but effective p of the logarithmic helix becomes a variable value at θ. The pitch p of the cylindrical and logarithmic helices are defined below:

· Cylindrical Helix:

$$p = \begin{cases} \theta R \tan \eta \ : \ 0 \ \leq \ \theta \ \leq \ 2\pi \\ 2\pi \ R \tan \eta \ : \ \theta \ > \ 2\pi \end{cases} \tag{2.4}$$

· Logarithmic Helix:

$$p = \begin{cases} L[1 - exp(a\theta)] \ : \ 0 \leq \theta \leq 2\pi \\ L[exp(-2a\pi) - 1]exp(a\theta) \ : \ \theta > 2\pi \end{cases} \tag{2.5}$$

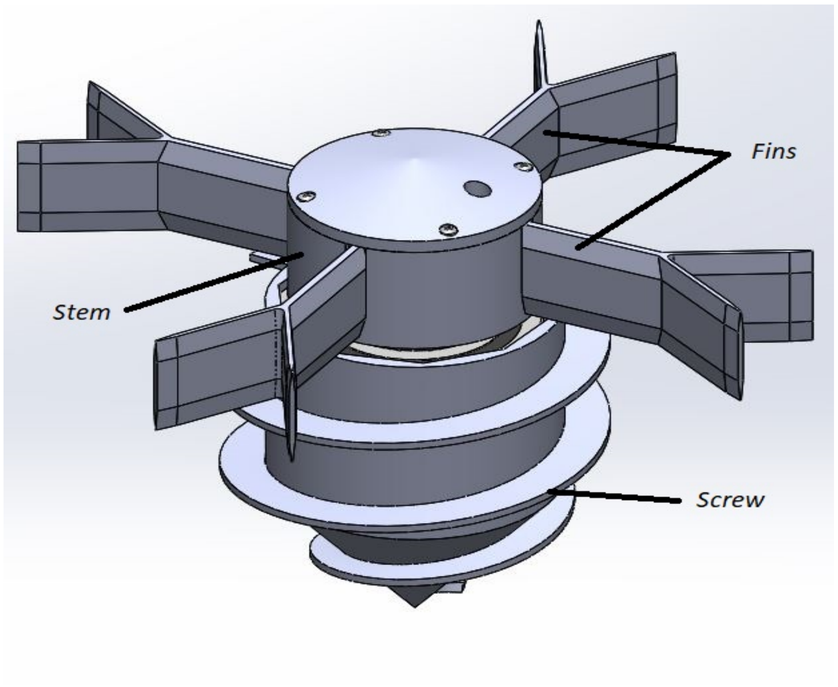
For the Archimedean screw models, the pitches are constrained by the screw length L at the initial point $\theta = 0$.

2.3 The Proposed Design

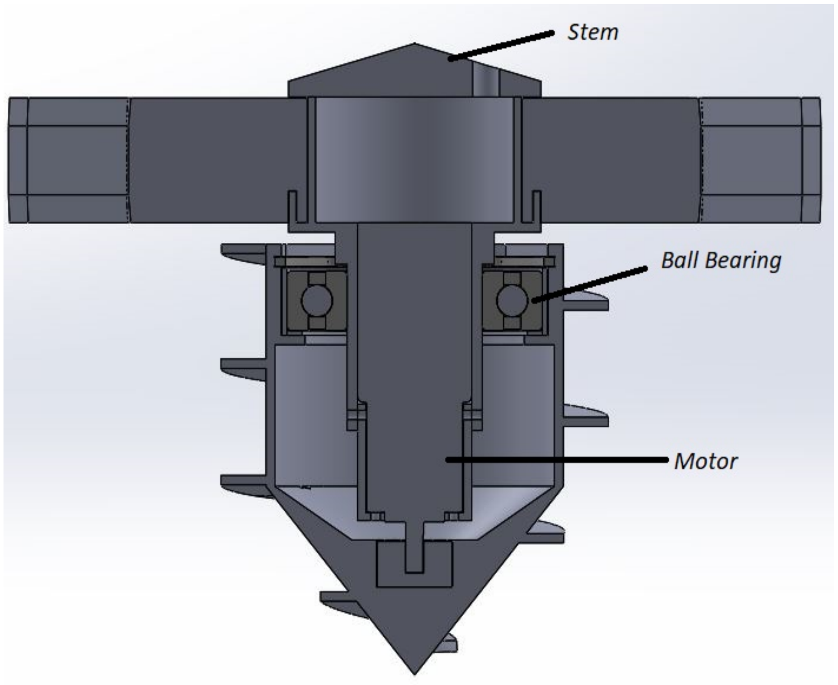
The proposed design consists of a screw design with both logarithmic and cylindrical helix, attached to a stem that houses the motor that drives the screw and the fins for stability when burrowing.

The robot is 4 inches long and 4 inches wide. The design consists of three major parts, namely, the main screw, the stem, and the fins. The main screw consists of a cylindrical helix screw which spans over a cylinder and a logarithmic helix that spans over a cone. A single screw simplifies mechanical designing eliminating places for failure. The stem is made up of a smaller hollow cylinder that fits into the main screw and larger hollow cylinder that houses the fins, power, and control unit of the robot. Similarly, the robot has four fins shaped in a Y form; they are fixed 90 degrees from each other. The fins and the stem is connected to the main screw by a ball bearing by press-fitting them together.

The inner radius of the screw is 46.50 mm, and the outer radius is 66.5 mm. The pitch of the logarithmic and cylindrical helix is 37 mm. The motor used was an 84 RPM HD premium planetary gear motor w/encoder, the 30A range current sensor ACS712 module current sensor was used to measure the current fed to the motor and Arduino Uno to control the robot. The ball bearing used is a sealed, trade No. 6209-2RS, for 45 mm shaft diameter.



(a) The Proposed Design



(b) Inner View of the Proposed Design

Figure 2.3: View of Proposed Design

2.4 Fin Design

The fin was designed to act like an anchor allowing the robot to travel down and up without being held. The fin has a large surface area to generate enough drag to prevent the robot from spinning when traveling. Previous designs have a top screw which spins opposite to the bottom screw to give the robot enough stability when traveling, but such designs require more power since those designs require two motors to operate. The fin is streamlined, thus decreasing drag in the vertical for smooth movement in and out of the medium. In summary, the fins allow the robot to travel without any external aid, hence allow the robot to autonomous.

2.5 Optimal Screw Design

In this paper, an optimal screw design is proposed as a method of improving the already proposed design. The optimal screw design consists of the same logarithmic and cylindrical helix design as the first but differs in the number of blades. In the optimal screw design the number of blades is increased to four(4) and the pitch also increased to 105 mm. This improvement was done because it has been seen that increasing the number of blades on an Archimedes screw improves the volume per turn ratio is increased and stability is also increased.[22].

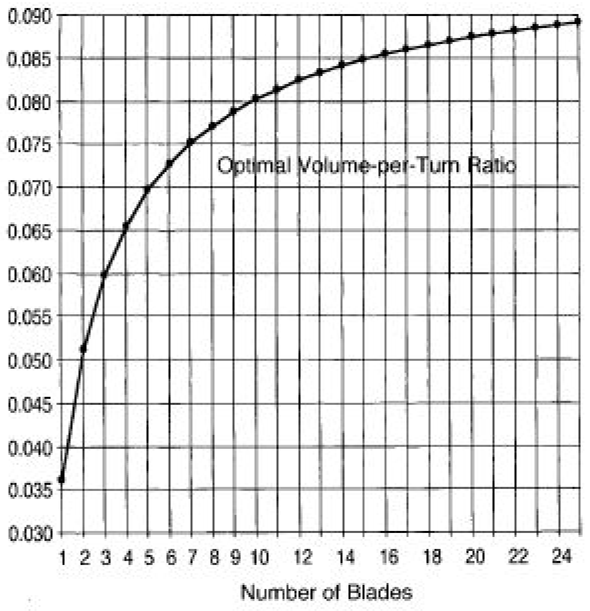


Figure 2.4: Optimal Volume per Turn Ratio Against Number of Blades[22]

2.6 Proportional-Integral-Derivative (PID) Controller

PID controller is a widely used control strategy because it is easily understandable and very effective. One can implement the control system without possessing a deep understanding of control theory. A PID captures the history of the system (through integration) and anticipates the future behaviour of the system (through differentiation). A PID controller was implemented in the design for the robot to enable an effective and easy control of the speed of the motor during travel in the medium.

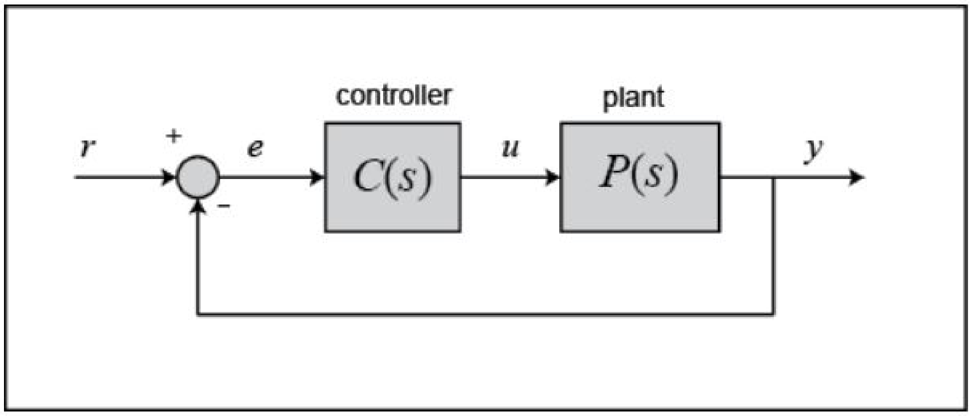


Figure 2.5: A Unity-Feedback System

A standard PID controller is also known as the "three-term" controller, whose transfer function is generally written in the the "parallel form" given by (2.6) or the "ideal form" given by (2.7)

$$G(s) = K_p + K_t\frac{1}{s} + K_{D^s} \tag{2.6}$$

$$= K_p(1 + \frac{1}{T_{I^s}} + T_{D^s}) \tag{2.7}$$

where K_p is the proportional gain, K_t the integral gain, K_D the derivative gain,

T_I the integral time constant and, T_D the derivative time constant. The "three-term" functionalities are highlighted by the following.

- The proportional term—providing an overall control action proportional to the error signal through the all–pass gain factor.
- The integral term—reducing steady-state errors through low-frequency compensation by an integrator.
- The derivative term—improving transient response through high-frequency compensation by a differentiator.[1][28]

The individual effects of these three terms on the closed-loop performance are summarized in Table 2.1.

Closed-Loop Response	**Rise Time**	**Overshoot**	**Settling Time**	**Steady-State Error**	**Stability**
Increasing K_P	Decrease	Increase	Small Increase	Decrease	Degrade
Increasing K_I	Small Decrease	Increase	Increase	Large Decrease	Degrade
Increasing K_D	Small Decrease	Decrease	Decrease	Minor Change	Improve

Table 2.1: Effects of Independent P, I and D Tuning

The PID control system that was deployed is represented in the chart below.

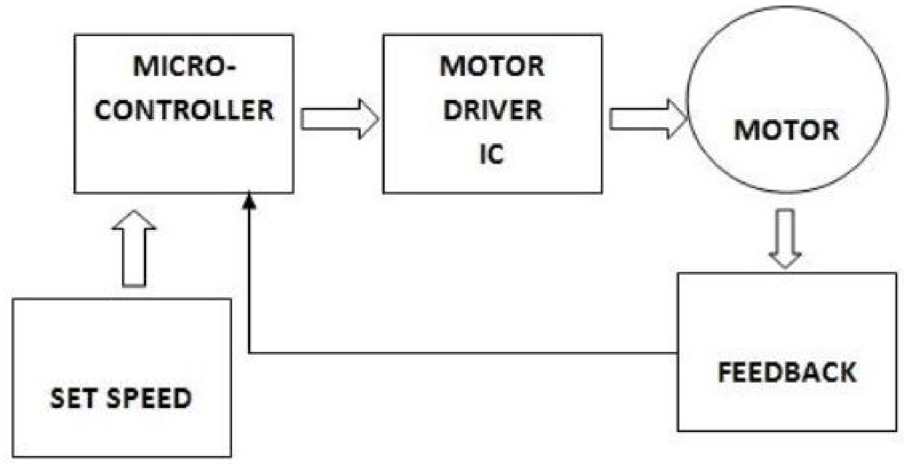


Figure 2.6: A Chart of the PID Control used in the Robot

2.7 Fabrication

The design was fabricated using an addictive manufacturing process, and PLA were used as the primary material. The individual parts were then assembled into the final product.

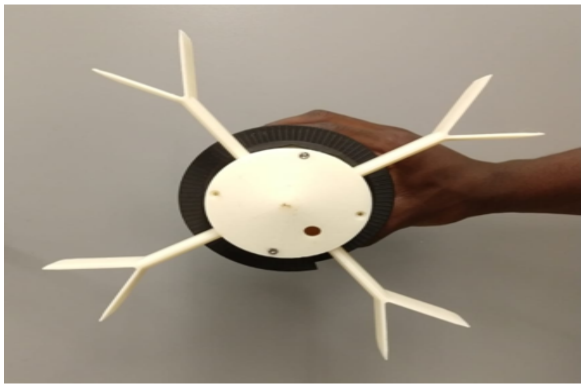


Figure 2.7: A Top View of the Sand Burrowing Robot

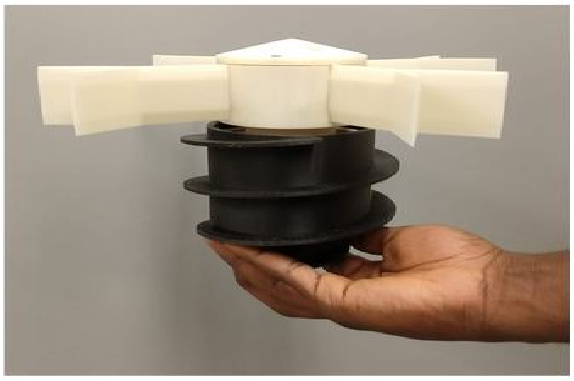


Figure 2.8: A Full Assembly of Sand Burrowing Robot

Chapter 3

EXPERIMENTAL SETUP AND PROCEDURE

3.1 Experimental Setup

In order to conduct an experiment to test the robot, an experimental setup to mimic favorable conditions was up together. For experiments, a 2mm glass beads were used as a medium of travel.Glass Beads are a round, spherical media that produce a softer and brighter finish angular medias[12]. A weight of 80 lbs of glass beads was used for experiment. The glass beads was poured into a 32-gallon container.Glass beads was used opposed to sand so that dynamic modelling simulation with glass beads of the same size for optimization purposes can be performed.

To capture the motion of the robot, the OptiTrack motion camera system was used. The OptiTrack system consists of two 1.3 MP resolution, with a wide 56^o field of view(FOV) and 120 FPS cameras.

The experimental setup consists of a DC Power supply, an OptiTrack Motion capture system to enable the movement of the robot to be recorded and a 32-gallon container that houses the glass beads, 12 inches deep.

3.2 Experimental Procedure

Firstly, The surface of the glass beads in the container is scraped and even out in the container. The robot is then placed in the container. The power is turned on for a short time so that the robot can adequately enter the media. The power is turned off when the top of the robot is on the same level as the top surface of the media. The glass beads are then even out again. Power fed to the robot from the power

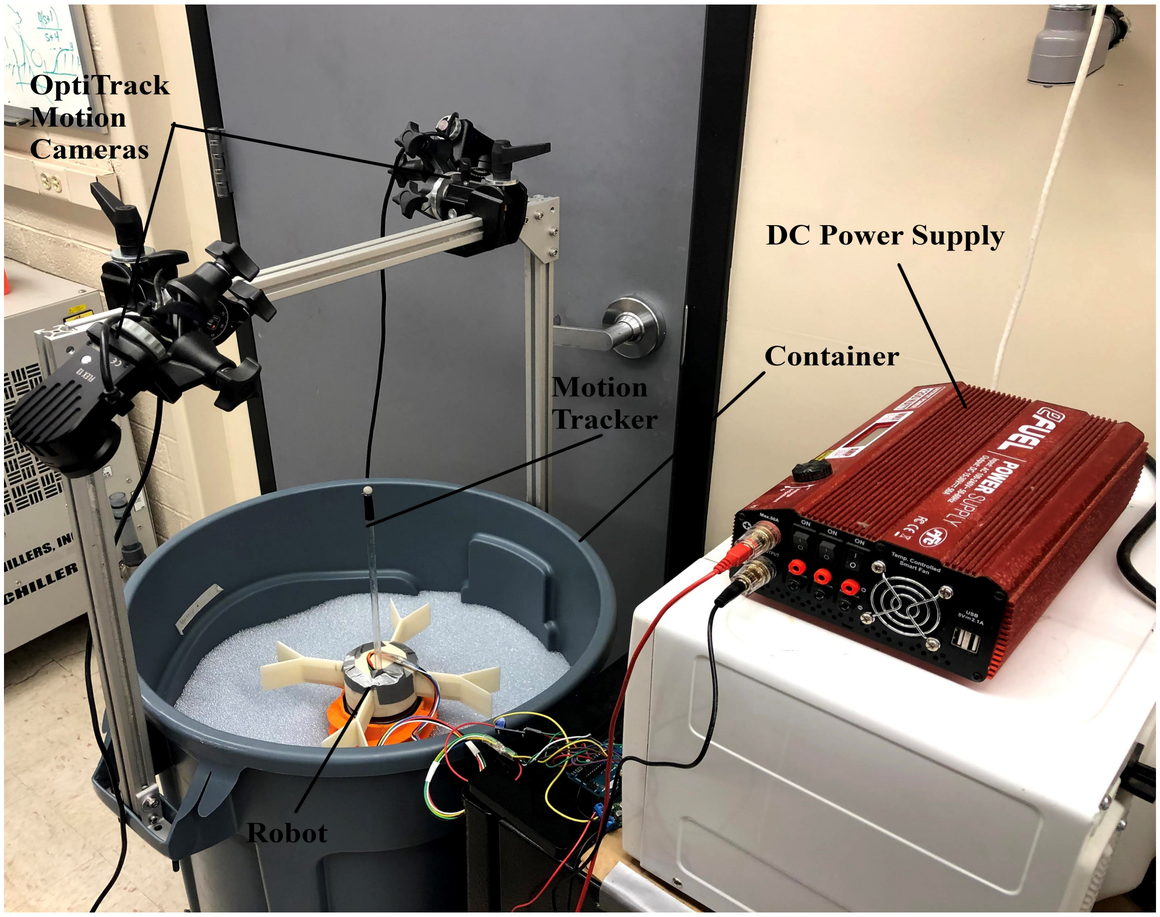


Figure 3.1: Experimental Setup

supply is 14.2 V; this is to ensure that an ample 12 V gets to the motor due to losses. Next, The OptiTrack motion capture cameras are turned on to begin recording the movement of the motion tracker, which is attached to an 8.5 inches long rod. The rod is connected to the top of the robot. This is to enable the motion capture system to track the robot when it is fully submerged in the medium. The speed and rotation of the robot are then set. The robot is controlled by an Arduino Uno; the Arduino is connected to a PC. The rotation of the motor is set to anticlockwise to enable the robot to travel down the medium and clockwise to enable it to travel to the surface of the medium. After the speed and the rotation of the motor is set, the power supply is turned on. The robot is allowed to travel down until about 95% of the motion tracker rod is submerged in the media, then the power supply is turned off. This is done to prevent the robot from hitting the bottom of the container. The OptiTrack's recording session is stopped and saved; the current readings from the Arduino is also saved. After the data has been saved the OptiTrack is turned on and the speed and rotation of the motor is set again. The power supply is turned off exactly when the motion tracker rod fully emerges out of the media .i.e., when the top surface of the robot is coincidental to the surface of the media. The data is then saved. The experiments were conducted at five(5) different speeds i.e. 20, 30, 40, 50 and 60 RPM's, and ten trials were conducted at each speed; five(5) trials for when the robot is traveling down the medium and the other five trials for when it is coming up. Successful trials were measured by the angle of the tilt of the motion tracker rod.

Two main experiments were conducted; one experiment had the robot using the proposed screw design, and the second experiment had the robot using the optimal screw design. The data from both experiments were collected and analyzed to establish if the optimal designed had the edge over the proposed design.

Chapter 4

RESULTS AND DISCUSSION

The data from the experiments were analyzed and plots were produced to visualize the data. The results are discussed in the following: :

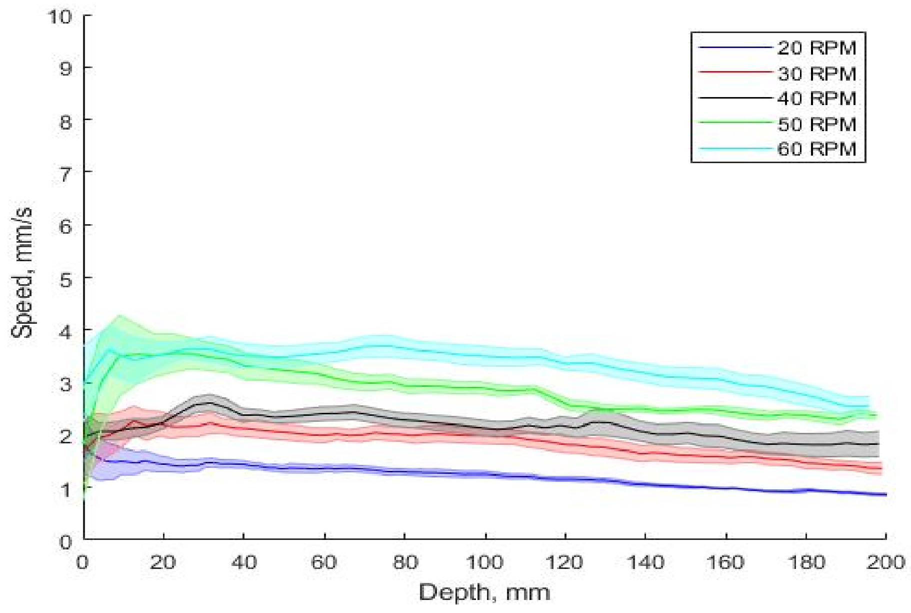


Figure 4.1: Plot of Speed against Depth for the Robot when Travelling In

The plot above shows the velocity against depth for each rpm. On the x-axis, the zero represents the surface of the media, and it increases till it reaches the bottom, which is 200 mm. It can deduce from the above plot that velocity decreases as the robot travels down the media. It follows Bernoulli's principle, which states that velocity decreases as pressure increases[16]. It is also seen that velocity increases as rpm increases; this is also expected since rpm is directly proportional to velocity.

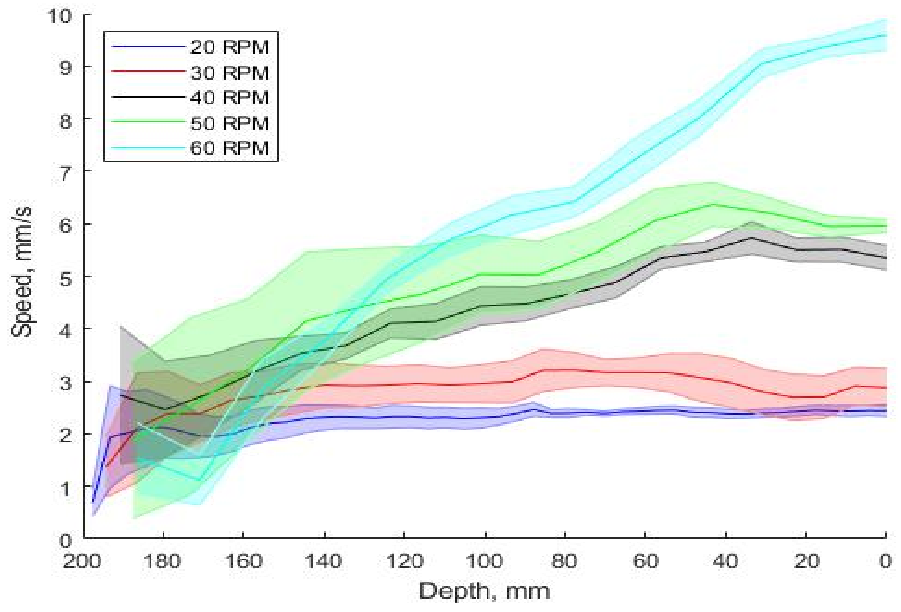


Figure 4.2: Plot of Speed against Depth for the Robot when Travelling Out

In figure 4.2, the plot of velocity against depth for each rpm when the craft is traveling out the media is presented. The x-axis presents the depth travelled with zero(0) representing the surface and -200 the bottom of the media. The plot shows that the velocity of the craft increases as it gets to the surface of the media, with 60 rpm recording the highest velocity. This is the reverse of the trend seen in figure 4.1, which is expected.

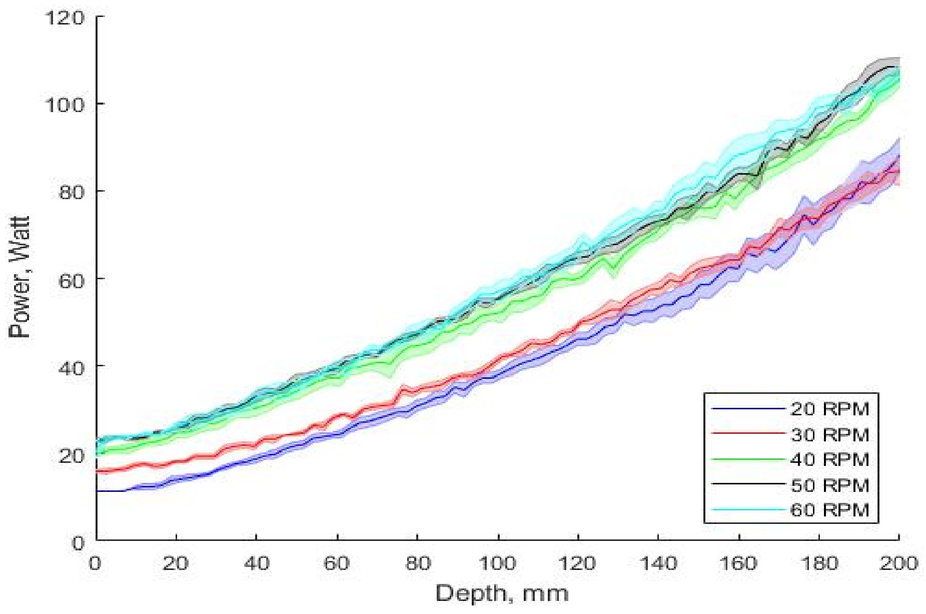


Figure 4.3: Plot of Power against Depth for the Robot when Traveling In

Figure 4.3 shows the plot of power against depth for traveling in the media. Again, zero(0) on the axis represents the surface of the media. The plot shows that the power drawn by the robot increase as the robot travels down the media. It is expected since the pressure exerted on the robot increases as the robot travels down.

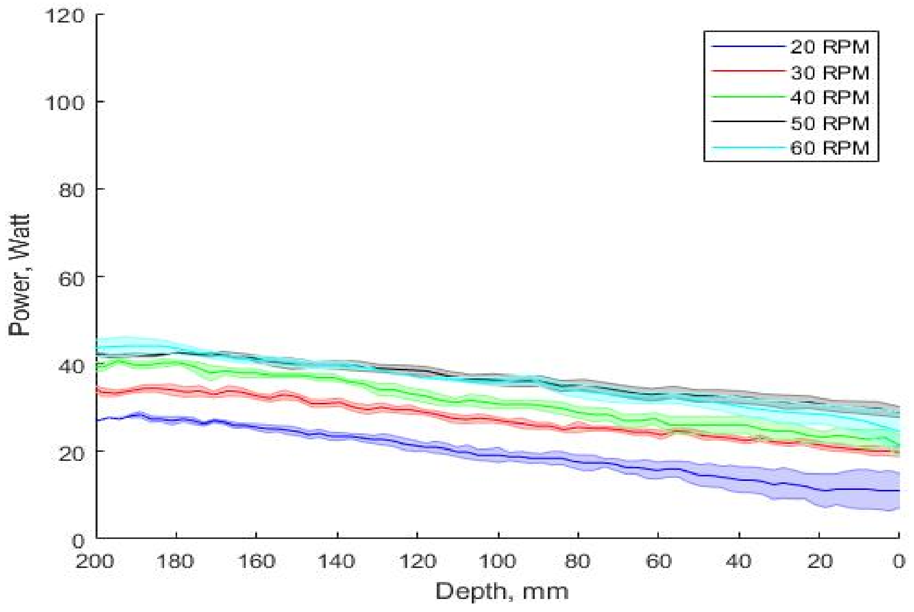


Figure 4.4: Plot of Power against Depth for the Robot when Traveling Out

The plot of power against depth when the robot is traveling out of the media is presented in figure 4.4. The plot shows the power increasing as it gets on the surface of the media. A careful observation of figure 4.3 and 4.4 reveals that even though the power is increasing in both graphs the maximum power drawn by the robot when traveling down is more than twice the maximum power dawn by the robot when traveling out.

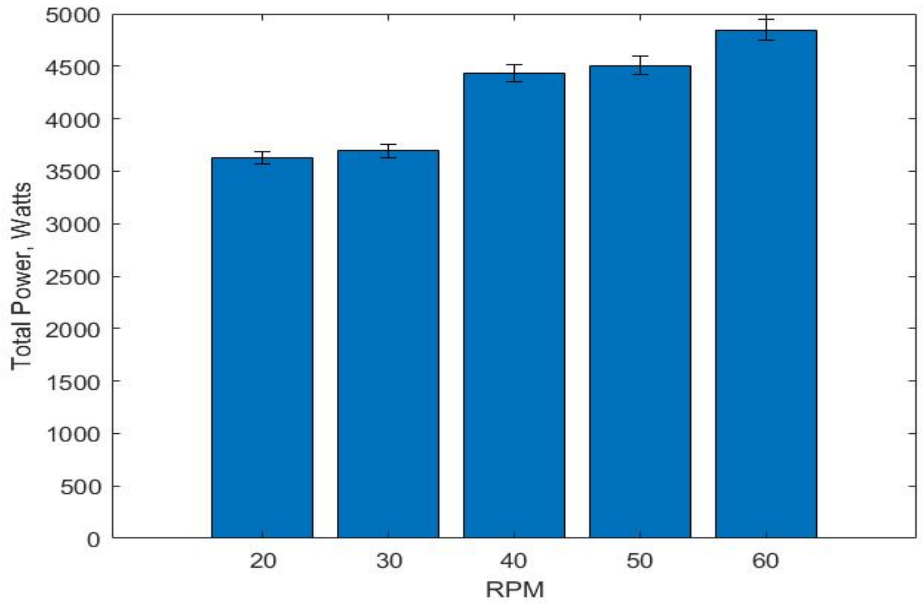


Figure 4.5: Plot of Total Power against RPM for the Robot when Traveling In

The plot of the total power against rpm when the robot is traveling in presented in figure 4.5 reveals an interesting fact. The total power was calculated by summing up the power drawn by the robot during travel. It can be deduced from the plot that 20 rpm draws the least amount of power when traveling down, even though 20 RPM, records the slowest velocity. It might be associated with the fact that at 20 rpm the motor records the maximum amount of torque compared to the other RPMs, as shown in figure 4.7. It enables the robot to travel through the media without drawing a lot of power. Hence, it can be said that 20 rpm is the most efficient speed when traveling down the media when speed is not of much importance.

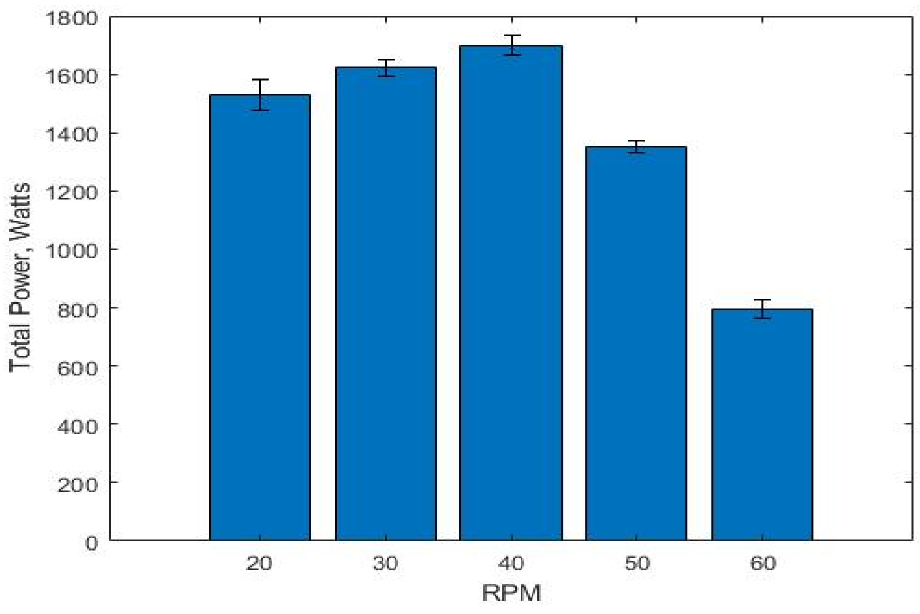


Figure 4.6: Plot of Total Power against RPM for the Robot when Travelling Out

In the plot of total power against rpm when the robot is traveling down the media, also shows an interesting trend. In the plot, it can be deduced that 60 rpm draws the least amount of power when traveling up the media. This is expected because at 60 rpm, the velocity is at its maximum hence it takes a relatively short time for the craft to get to the surface hence drawing a small amount of power. Therefore, 60 rpm represents the most efficient speed when traveling up the media.

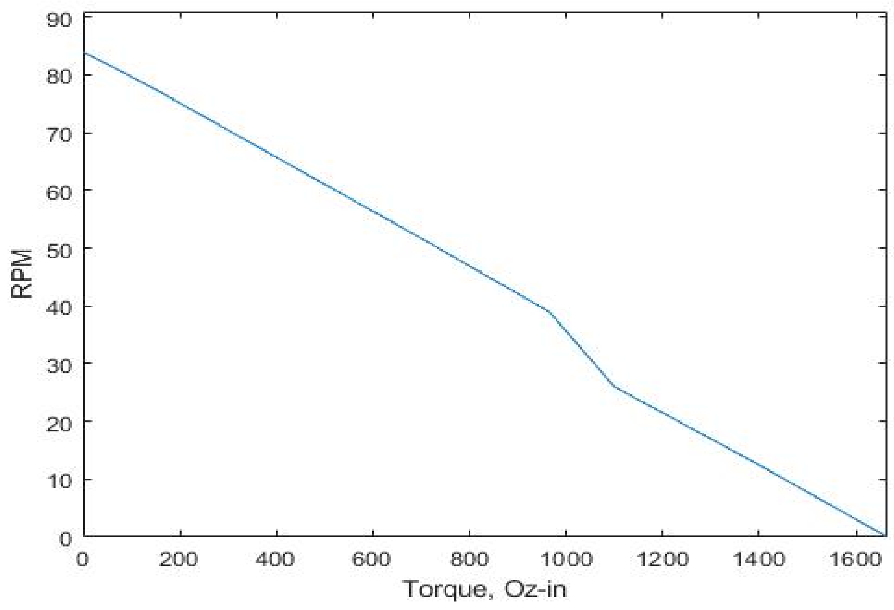


Figure 4.7: Plot of RPM against Torque of the Motor

The figure shows that speed of the motor is inversely proportional to the torque. Hence, as the torque of the motor increases as the speed decreases.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This thesis has summarized the effort to design, fabricate, and characterize sand burrowing robot. The robot consists of a screw with fins, and the fins act as an anchor enabling the robot to travel through the medium. An optimal screw design was also designed and fabricated. Experiments were performed on the proposed screw and is to be performed on the optimal screw design. The data from the proposed screw has been analyzed and will be compared to the analyzed data when experiments are performed on the optimal screw. It is projected that the optimal screw design will perform better than the proposed screw through the analysis made in chapter 2. Overall, the research is considered as a success.

5.2 Future Work

Even though the sand burrowing robot was designed, fabricated, and characterized, there is still much development that can be implemented to better it. Some future development that can help improve the overall design is discussed below;

During experiments, it was identified that after a few trials the motor heated up, and hence there was the need to "make it rest" before further trials can be undertaken. This is due to the fact that the robot is enclosed. This prevents the air from coming in or going out, causing the robot to heat up. One suggestion is to used metal over PLA as the build material. Since metal is a good conductor of heat, the metal will conduct the heat and dissipate it to the surrounding bodies. Also, a cooling system can be integrated into the device.

Traveling up and down the medium is suitable but not sufficient for complex subsurface exploration. Hence, One suggestion is to have an active design over static fins. Using active fins will enable the robot to change direction when in the medium. This will allow for complex subsurface exploration, which usually requires complex movement.

Stability is another area that needs improvement. Even though the fins do a great job at anchoring the robot, the design of the screw is best suited to traveling down the medium but not coming up with the medium. This causes the robot to come out of the medium at an angle greater than zero when using the vertical axis as the reference.

REFERENCES

[1] Ang, K. H., G. Chong and Y. Li, "Pid control system analysis, design, and technology", IEEE transactions on control systems technology **13**, 4, 559–576 (2005).

[2] Archimedes, "Archimedes' screw.", URL `https://hr.m.wikipedia.org/wiki/Datoteka:Archime` [Online; accessed April 8, 2020] (2007).

[3] Bock, T., "Construction robotics", Autonomous Robots **22**, 3, 201–209 (2007).

[4] Darukhanavala, C., A. Lycas, A. Mittal and A. Suresh, "Design of a bimodal self-burying robot", in "2013 IEEE International Conference on Robotics and Automation", pp. 5600–5605 (IEEE, 2013).

[5] Freeman, P. W. and H. H. Genoways, "Recent northern records of the nine-banded armadillo (dasypodidae) in nebraska", The Southwestern Naturalist pp. 491–495 (1998).

[6] Heiken, G. H., D. T. Vaniman and B. M. French, "Lunar sourcebook-a user's guide to the moon", Research supported by NASA,. Cambridge, England, Cambridge University Press, 1991, 753 p. No individual items are abstracted in this volume. (1991).

[7] Hollingum, J., "Robots in agriculture", Industrial Robot: An International Journal (1999).

[8] Huang, S., Y. Tang, H. Bagheri, D. Li, A. Ardente, D. Aukes, H. Marvi and J. J. Tao, "Effects of friction anisotropy on upward burrowing behavior of soft robots in granular materials", Advanced Intelligent Systems (2020).

[9] Huang, S., Y. Tang *et al.*, "Sbor: a minimalistic soft self-burrowing-out robot inspired by razor clams", Bioinspiration & Biomimetics (2020).

[10] Ji, W., D. Chen, H. Jia and J. Tong, "Experimental investigation into soil-cutting performance of the claws of mole rat (scaptochirus moschatus)", Journal of Bionic Engineering **7**, S166–S171 (2010).

[11] Kenneth Catania, Vanderbilt University, "A photograph of scalopus aquaticus", URL `https://www.nsf.gov/news/mmg/mmg_disp.jsp?med_id=64599&amp;from=img`, [Online; accessed April 8, 2020] (2008).

[12] Kramer Industries Inc, "Our product catalog", URL `https://www.kramerindustriesonline.com/kramershop/`, [Online; accessed April 8, 2020] (2008).

[13] Lashofer, A., W. Hawle and B. Pelikan, "State of technology and design guidelines for the archimedes screw turbine", Univ. Nat. Resour. Life Sci. Vienna pp. 1–8 (2012).

[14] Macdonald, D. W., *The encyclopedia of mammals* (Oxford University Press, 2010).

[15] Miklosovic, D., M. Murray, L. Howle and F. Fish, "Leading-edge tubercles delay stall on humpback whale (megaptera novaeangliae) flippers", Physics of fluids **16**, 5, L39–L42 (2004).

[16] Munson, B. R., T. H. Okiishi, W. W. Huebsch and A. P. Rothmayer, *Fluid mechanics* (Wiley Singapore, 2013).

[17] Nagaoka, K. and T. Kubota, "Analytic study on screw drilling mechanism", in "2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)", pp. 1579–1584 (IEEE, 2009).

[18] Nagaoka, K., T. Kubota, M. Otsuki and S. Tanaka, "Experimental analysis of a screw drilling mechanism for lunar robotic subsurface exploration", Advanced Robotics **24**, 8-9, 1127–1147 (2010).

[19] Nakamura, Y. and R. Mukherjee, "Nonholonomic path planning of space robots via bi-directional approach", in "Proceedings., IEEE International Conference on Robotics and Automation", pp. 1764–1769 (IEEE, 1990).

[20] Nordstrom, K., D. Dorsch, W. Losert and V. AG Winter, "Microstructural view of burrowing with a bioinspired digging robot", Physical Review E **92**, 4, 042204 (2015).

[21] Primicerio, J., S. F. Di Gennaro, E. Fiorillo, L. Genesio, E. Lugato, A. Matese and F. P. Vaccari, "A flexible unmanned aerial vehicle for precision agriculture", Precision Agriculture **13**, 4, 517–523 (2012).

[22] Rorres, C., "The turn of the screw: Optimal design of an archimedes screw", Journal of hydraulic engineering **126**, 1, 72–80 (2000).

[23] Stoker, C., A. Gonzales and J. Zavaleta, "Moon/mars underground mole", in "Proc. 2007 NASA Science Technology Conf", vol. 6 (2007).

[24] Tao, J. S. Huang and Y. Tang, "Bioinspired self-burrowing-out robot in dry sand", Journal of Geotechnical and Geoenvironmental Engineering **145**, 12, 02819002 (2019).

[25] Texier, B. D., A. Ibarra and F. Melo, "Helical locomotion in a granular medium", Physical review letters **119**, 6, 068003 (2017).

[26] Trueman, E., "The dynamics of burrowing in ensis (bivalvia)", Proceedings of the Royal Society of London. Series B. Biological Sciences **166**, 1005, 459–476 (1967).

[27] Valdés, R., V. Angeles, E. de la Calleja and R. Zenit, "Self-propulsion of a helical swimmer in granular matter", Physical Review Fluids **4**, 8, 084302 (2019).

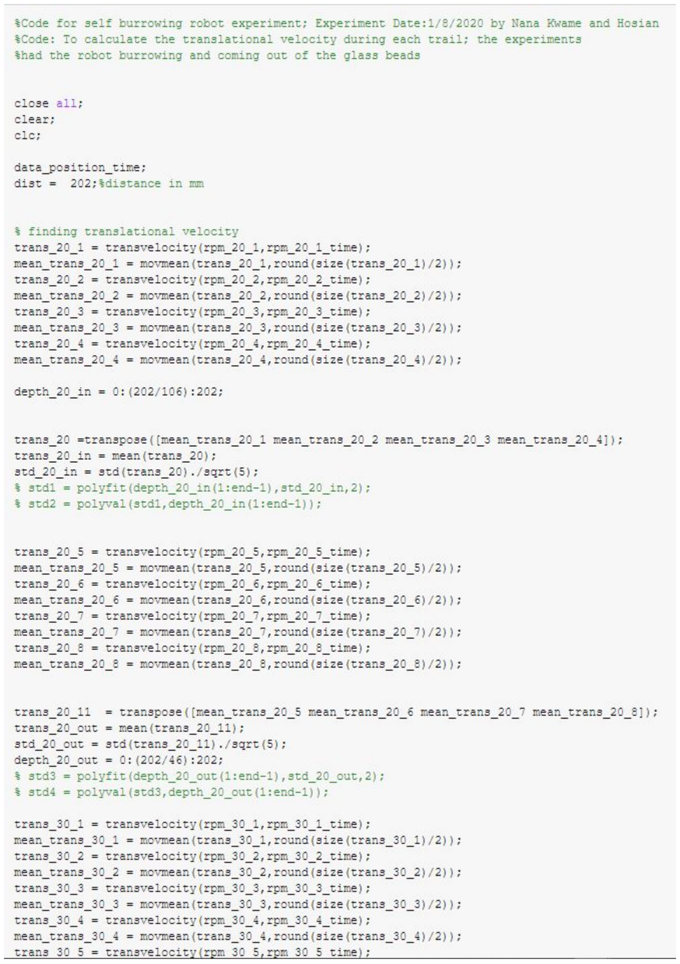
[28] Wang, Q.-G., T.-H. Lee, H.-W. Fung, Q. Bi and Y. Zhang, "Pid tuning for improved performance", IEEE Transactions on control systems technology **7**, 4, 457–465 (1999).

[29] Watanabe, K., S. Shimoda, T. Kubota and I. Nakatani, "A mole-type drilling robot for lunar subsurface exploration", in "Proceeding of the 7th International Symposium on Artificial Intelligence and Robotics & Automation in Space", (2003).

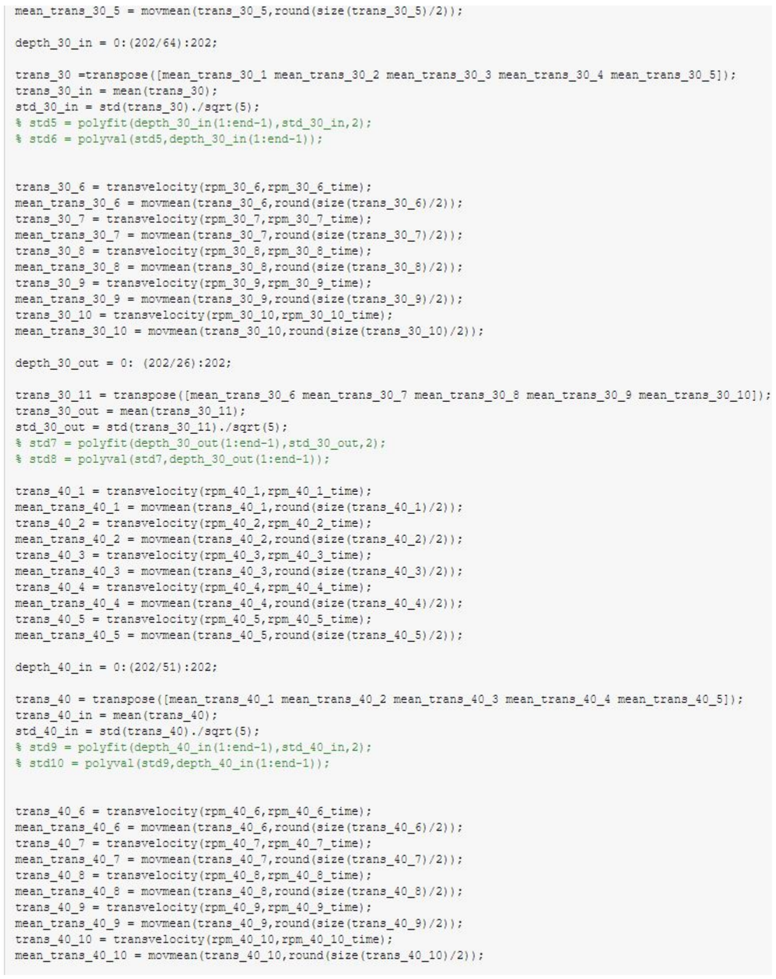
[30] Winter, A. G., R. L. Deits and A. E. Hosoi, "Localized fluidization burrowing mechanics of ensis directus", Journal of experimental biology **215**, 12, 2072–2080 (2012).

APPENDIX A

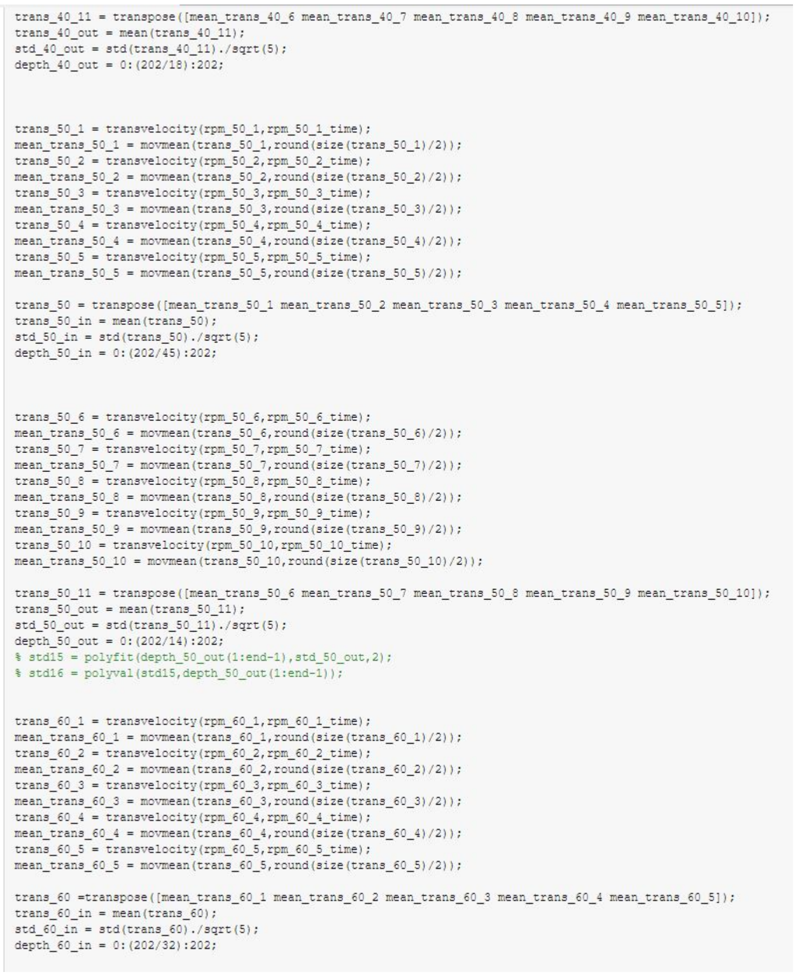
MATLAB CODES



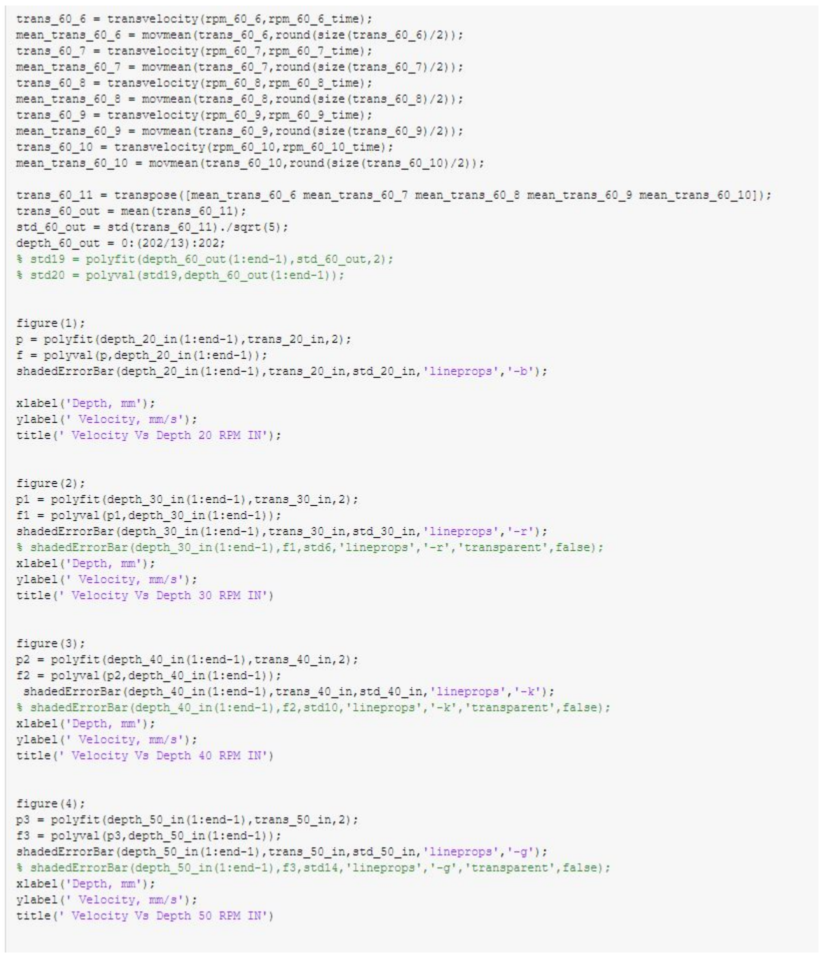
```
%Code for self burrowing robot experiment; Experiment Date:1/8/2020 by Nana Kwame and Hosian
%Code: To calculate the translational velocity during each trail; the experiments
%had the robot burrowing and coming out of the glass beads


close all;
clear;
clc;

data_position_time;
dist =  202;%distance in mm


% finding translational velocity
trans_20_1 = transvelocity(rpm_20_1,rpm_20_1_time);
mean_trans_20_1 = movmean(trans_20_1,round(size(trans_20_1)/2));
trans_20_2 = transvelocity(rpm_20_2,rpm_20_2_time);
mean_trans_20_2 = movmean(trans_20_2,round(size(trans_20_2)/2));
trans_20_3 = transvelocity(rpm_20_3,rpm_20_3_time);
mean_trans_20_3 = movmean(trans_20_3,round(size(trans_20_3)/2));
trans_20_4 = transvelocity(rpm_20_4,rpm_20_4_time);
mean_trans_20_4 = movmean(trans_20_4,round(size(trans_20_4)/2));

depth_20_in = 0:(202/106):202;


trans_20 =transpose([mean_trans_20_1 mean_trans_20_2 mean_trans_20_3 mean_trans_20_4]);
trans_20_in = mean(trans_20);
std_20_in = std(trans_20)./sqrt(5);
% std1 = polyfit(depth_20_in(1:end-1),std_20_in,2);
% std2 = polyval(std1,depth_20_in(1:end-1));


trans_20_5 = transvelocity(rpm_20_5,rpm_20_5_time);
mean_trans_20_5 = movmean(trans_20_5,round(size(trans_20_5)/2));
trans_20_6 = transvelocity(rpm_20_6,rpm_20_6_time);
mean_trans_20_6 = movmean(trans_20_6,round(size(trans_20_6)/2));
trans_20_7 = transvelocity(rpm_20_7,rpm_20_7_time);
mean_trans_20_7 = movmean(trans_20_7,round(size(trans_20_7)/2));
trans_20_8 = transvelocity(rpm_20_8,rpm_20_8_time);
mean_trans_20_8 = movmean(trans_20_8,round(size(trans_20_8)/2));


trans_20_11  = transpose([mean_trans_20_5 mean_trans_20_6 mean_trans_20_7 mean_trans_20_8]);
trans_20_out = mean(trans_20_11);
std_20_out = std(trans_20_11)./sqrt(5);
depth_20_out = 0:(202/46):202;
% std3 = polyfit(depth_20_out(1:end-1),std_20_out,2);
% std4 = polyval(std3,depth_20_out(1:end-1));

trans_30_1 = transvelocity(rpm_30_1,rpm_30_1_time);
mean_trans_30_1 = movmean(trans_30_1,round(size(trans_30_1)/2));
trans_30_2 = transvelocity(rpm_30_2,rpm_30_2_time);
mean_trans_30_2 = movmean(trans_30_2,round(size(trans_30_2)/2));
trans_30_3 = transvelocity(rpm_30_3,rpm_30_3_time);
mean_trans_30_3 = movmean(trans_30_3,round(size(trans_30_3)/2));
trans_30_4 = transvelocity(rpm_30_4,rpm_30_4_time);
mean_trans_30_4 = movmean(trans_30_4,round(size(trans_30_4)/2));
trans_30_5 = transvelocity(rpm_30_5,rpm_30_5_time);
```



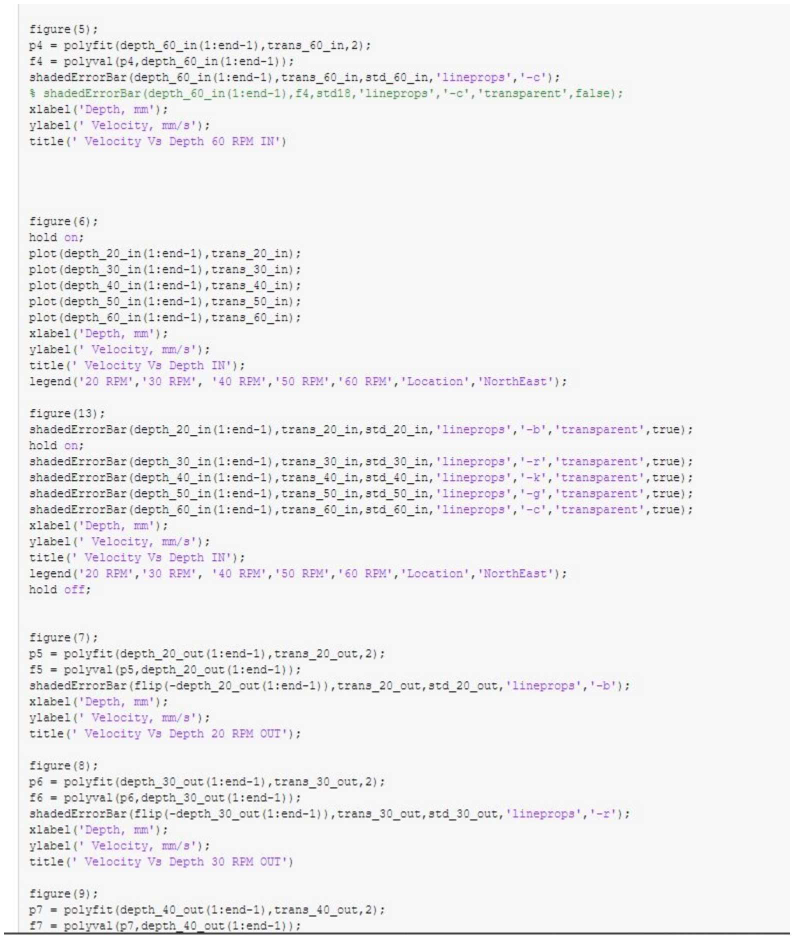
```
mean_trans_30_5 = movmean(trans_30_5,round(size(trans_30_5)/2));

depth_30_in = 0:(202/64):202;

trans_30 =transpose([mean_trans_30_1 mean_trans_30_2 mean_trans_30_3 mean_trans_30_4 mean_trans_30_5]);
trans_30_in = mean(trans_30);
std_30_in = std(trans_30)./sqrt(5);
% std5 = polyfit(depth_30_in(1:end-1),std_30_in,2);
% std6 = polyval(std5,depth_30_in(1:end-1));


trans_30_6 = transvelocity(rpm_30_6,rpm_30_6_time);
mean_trans_30_6 = movmean(trans_30_6,round(size(trans_30_6)/2));
trans_30_7 = transvelocity(rpm_30_7,rpm_30_7_time);
mean_trans_30_7 = movmean(trans_30_7,round(size(trans_30_7)/2));
trans_30_8 = transvelocity(rpm_30_8,rpm_30_8_time);
mean_trans_30_8 = movmean(trans_30_8,round(size(trans_30_8)/2));
trans_30_9 = transvelocity(rpm_30_9,rpm_30_9_time);
mean_trans_30_9 = movmean(trans_30_9,round(size(trans_30_9)/2));
trans_30_10 = transvelocity(rpm_30_10,rpm_30_10_time);
mean_trans_30_10 = movmean(trans_30_10,round(size(trans_30_10)/2));

depth_30_out = 0: (202/26):202;

trans_30_11 = transpose([mean_trans_30_6 mean_trans_30_7 mean_trans_30_8 mean_trans_30_9 mean_trans_30_10]);
trans_30_out = mean(trans_30_11);
std_30_out = std(trans_30_11)./sqrt(5);
% std7 = polyfit(depth_30_out(1:end-1),std_30_out,2);
% std8 = polyval(std7,depth_30_out(1:end-1));

trans_40_1 = transvelocity(rpm_40_1,rpm_40_1_time);
mean_trans_40_1 = movmean(trans_40_1,round(size(trans_40_1)/2));
trans_40_2 = transvelocity(rpm_40_2,rpm_40_2_time);
mean_trans_40_2 = movmean(trans_40_2,round(size(trans_40_2)/2));
trans_40_3 = transvelocity(rpm_40_3,rpm_40_3_time);
mean_trans_40_3 = movmean(trans_40_3,round(size(trans_40_3)/2));
trans_40_4 = transvelocity(rpm_40_4,rpm_40_4_time);
mean_trans_40_4 = movmean(trans_40_4,round(size(trans_40_4)/2));
trans_40_5 = transvelocity(rpm_40_5,rpm_40_5_time);
mean_trans_40_5 = movmean(trans_40_5,round(size(trans_40_5)/2));

depth_40_in = 0:(202/51):202;

trans_40 = transpose([mean_trans_40_1 mean_trans_40_2 mean_trans_40_3 mean_trans_40_4 mean_trans_40_5]);
trans_40_in = mean(trans_40);
std_40_in = std(trans_40)./sqrt(5);
% std9 = polyfit(depth_40_in(1:end-1),std_40_in,2);
% std10 = polyval(std9,depth_40_in(1:end-1));


trans_40_6 = transvelocity(rpm_40_6,rpm_40_6_time);
mean_trans_40_6 = movmean(trans_40_6,round(size(trans_40_6)/2));
trans_40_7 = transvelocity(rpm_40_7,rpm_40_7_time);
mean_trans_40_7 = movmean(trans_40_7,round(size(trans_40_7)/2));
trans_40_8 = transvelocity(rpm_40_8,rpm_40_8_time);
mean_trans_40_8 = movmean(trans_40_8,round(size(trans_40_8)/2));
trans_40_9 = transvelocity(rpm_40_9,rpm_40_9_time);
mean_trans_40_9 = movmean(trans_40_9,round(size(trans_40_9)/2));
trans_40_10 = transvelocity(rpm_40_10,rpm_40_10_time);
mean_trans_40_10 = movmean(trans_40_10,round(size(trans_40_10)/2));
```



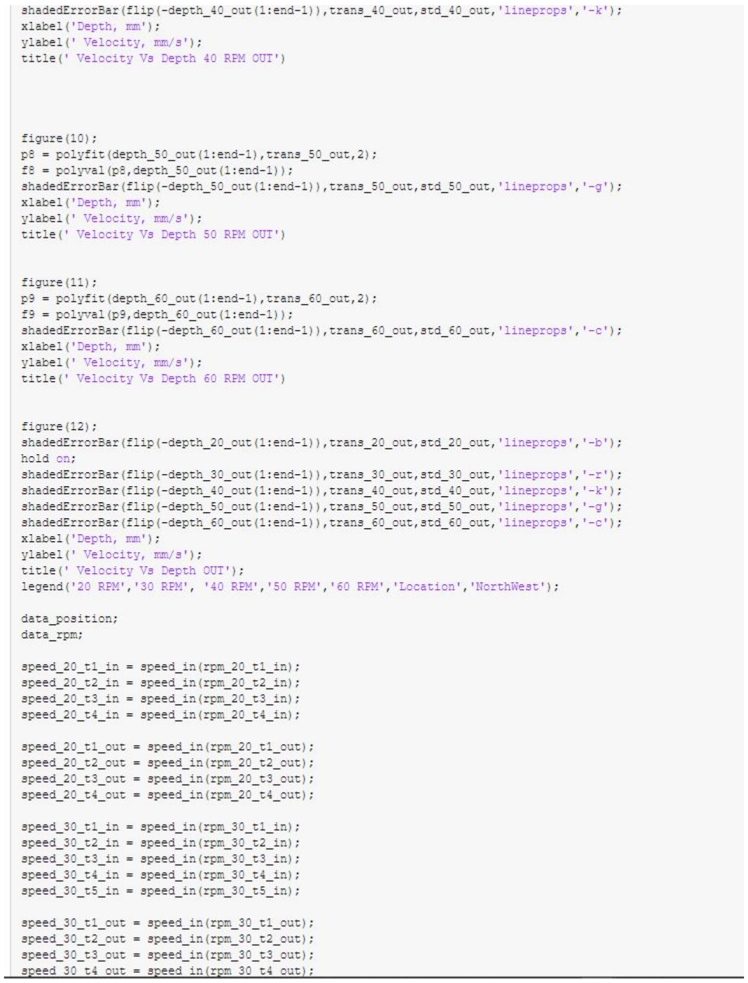
```
trans_40_11 = transpose([mean_trans_40_6 mean_trans_40_7 mean_trans_40_8 mean_trans_40_9 mean_trans_40_10]);
trans_40_out = mean(trans_40_11);
std_40_out = std(trans_40_11)./sqrt(5);
depth_40_out = 0:(202/18):202;



trans_50_1 = transvelocity(rpm_50_1,rpm_50_1_time);
mean_trans_50_1 = movmean(trans_50_1,round(size(trans_50_1)/2));
trans_50_2 = transvelocity(rpm_50_2,rpm_50_2_time);
mean_trans_50_2 = movmean(trans_50_2,round(size(trans_50_2)/2));
trans_50_3 = transvelocity(rpm_50_3,rpm_50_3_time);
mean_trans_50_3 = movmean(trans_50_3,round(size(trans_50_3)/2));
trans_50_4 = transvelocity(rpm_50_4,rpm_50_4_time);
mean_trans_50_4 = movmean(trans_50_4,round(size(trans_50_4)/2));
trans_50_5 = transvelocity(rpm_50_5,rpm_50_5_time);
mean_trans_50_5 = movmean(trans_50_5,round(size(trans_50_5)/2));

trans_50 = transpose([mean_trans_50_1 mean_trans_50_2 mean_trans_50_3 mean_trans_50_4 mean_trans_50_5]);
trans_50_in = mean(trans_50);
std_50_in = std(trans_50)./sqrt(5);
depth_50_in = 0:(202/45):202;



trans_50_6 = transvelocity(rpm_50_6,rpm_50_6_time);
mean_trans_50_6 = movmean(trans_50_6,round(size(trans_50_6)/2));
trans_50_7 = transvelocity(rpm_50_7,rpm_50_7_time);
mean_trans_50_7 = movmean(trans_50_7,round(size(trans_50_7)/2));
trans_50_8 = transvelocity(rpm_50_8,rpm_50_8_time);
mean_trans_50_8 = movmean(trans_50_8,round(size(trans_50_8)/2));
trans_50_9 = transvelocity(rpm_50_9,rpm_50_9_time);
mean_trans_50_9 = movmean(trans_50_9,round(size(trans_50_9)/2));
trans_50_10 = transvelocity(rpm_50_10,rpm_50_10_time);
mean_trans_50_10 = movmean(trans_50_10,round(size(trans_50_10)/2));

trans_50_11 = transpose([mean_trans_50_6 mean_trans_50_7 mean_trans_50_8 mean_trans_50_9 mean_trans_50_10]);
trans_50_out = mean(trans_50_11);
std_50_out = std(trans_50_11)./sqrt(5);
depth_50_out = 0:(202/14):202;
% std15 = polyfit(depth_50_out(1:end-1),std_50_out,2);
% std16 = polyval(std15,depth_50_out(1:end-1));


trans_60_1 = transvelocity(rpm_60_1,rpm_60_1_time);
mean_trans_60_1 = movmean(trans_60_1,round(size(trans_60_1)/2));
trans_60_2 = transvelocity(rpm_60_2,rpm_60_2_time);
mean_trans_60_2 = movmean(trans_60_2,round(size(trans_60_2)/2));
trans_60_3 = transvelocity(rpm_60_3,rpm_60_3_time);
mean_trans_60_3 = movmean(trans_60_3,round(size(trans_60_3)/2));
trans_60_4 = transvelocity(rpm_60_4,rpm_60_4_time);
mean_trans_60_4 = movmean(trans_60_4,round(size(trans_60_4)/2));
trans_60_5 = transvelocity(rpm_60_5,rpm_60_5_time);
mean_trans_60_5 = movmean(trans_60_5,round(size(trans_60_5)/2));

trans_60 =transpose([mean_trans_60_1 mean_trans_60_2 mean_trans_60_3 mean_trans_60_4 mean_trans_60_5]);
trans_60_in = mean(trans_60);
std_60_in = std(trans_60)./sqrt(5);
depth_60_in = 0:(202/32):202;
```



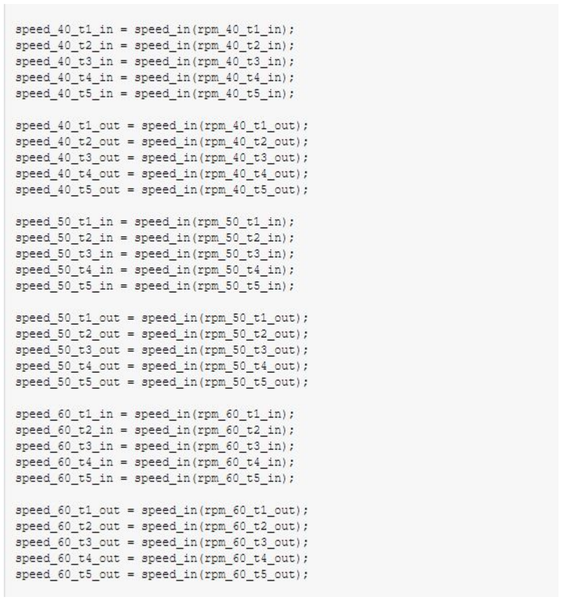
```
trans_60_6 = transvelocity(rpm_60_6,rpm_60_6_time);
mean_trans_60_6 = movmean(trans_60_6,round(size(trans_60_6)/2));
trans_60_7 = transvelocity(rpm_60_7,rpm_60_7_time);
mean_trans_60_7 = movmean(trans_60_7,round(size(trans_60_7)/2));
trans_60_8 = transvelocity(rpm_60_8,rpm_60_8_time);
mean_trans_60_8 = movmean(trans_60_8,round(size(trans_60_8)/2));
trans_60_9 = transvelocity(rpm_60_9,rpm_60_9_time);
mean_trans_60_9 = movmean(trans_60_9,round(size(trans_60_9)/2));
trans_60_10 = transvelocity(rpm_60_10,rpm_60_10_time);
mean_trans_60_10 = movmean(trans_60_10,round(size(trans_60_10)/2));

trans_60_11 = transpose([mean_trans_60_6 mean_trans_60_7 mean_trans_60_8 mean_trans_60_9 mean_trans_60_10]);
trans_60_out = mean(trans_60_11);
std_60_out = std(trans_60_11)./sqrt(5);
depth_60_out = 0:(202/13):202;
% std19 = polyfit(depth_60_out(1:end-1),std_60_out,2);
% std20 = polyval(std19,depth_60_out(1:end-1));


figure(1);
p = polyfit(depth_20_in(1:end-1),trans_20_in,2);
f = polyval(p,depth_20_in(1:end-1));
shadedErrorBar(depth_20_in(1:end-1),trans_20_in,std_20_in,'lineprops','-b');

xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth 20 RPM IN');


figure(2);
p1 = polyfit(depth_30_in(1:end-1),trans_30_in,2);
f1 = polyval(p1,depth_30_in(1:end-1));
shadedErrorBar(depth_30_in(1:end-1),trans_30_in,std_30_in,'lineprops','-r');
% shadedErrorBar(depth_30_in(1:end-1),f1,std6,'lineprops','-r','transparent',false);
xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth 30 RPM IN')


figure(3);
p2 = polyfit(depth_40_in(1:end-1),trans_40_in,2);
f2 = polyval(p2,depth_40_in(1:end-1));
 shadedErrorBar(depth_40_in(1:end-1),trans_40_in,std_40_in,'lineprops','-k');
% shadedErrorBar(depth_40_in(1:end-1),f2,std10,'lineprops','-k','transparent',false);
xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth 40 RPM IN')


figure(4);
p3 = polyfit(depth_50_in(1:end-1),trans_50_in,2);
f3 = polyval(p3,depth_50_in(1:end-1));
shadedErrorBar(depth_50_in(1:end-1),trans_50_in,std_50_in,'lineprops','-g');
% shadedErrorBar(depth_50_in(1:end-1),f3,std14,'lineprops','-g','transparent',false);
xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth 50 RPM IN')
```



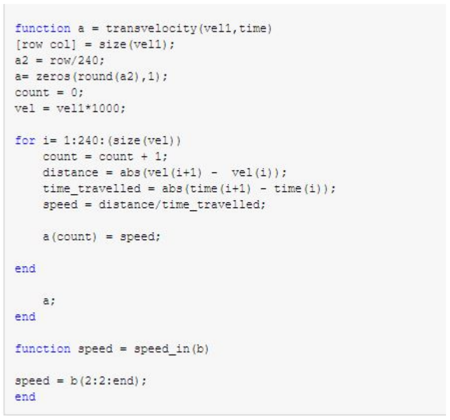
```
figure(5);
p4 = polyfit(depth_60_in(1:end-1),trans_60_in,2);
f4 = polyval(p4,depth_60_in(1:end-1));
shadedErrorBar(depth_60_in(1:end-1),trans_60_in,std_60_in,'lineprops','-c');
% shadedErrorBar(depth_60_in(1:end-1),f4,std18,'lineprops','-c','transparent',false);
xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth 60 RPM IN')




figure(6);
hold on;
plot(depth_20_in(1:end-1),trans_20_in);
plot(depth_30_in(1:end-1),trans_30_in);
plot(depth_40_in(1:end-1),trans_40_in);
plot(depth_50_in(1:end-1),trans_50_in);
plot(depth_60_in(1:end-1),trans_60_in);
xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth IN');
legend('20 RPM','30 RPM', '40 RPM','50 RPM','60 RPM','Location','NorthEast');

figure(13);
shadedErrorBar(depth_20_in(1:end-1),trans_20_in,std_20_in,'lineprops','-b','transparent',true);
hold on;
shadedErrorBar(depth_30_in(1:end-1),trans_30_in,std_30_in,'lineprops','-r','transparent',true);
shadedErrorBar(depth_40_in(1:end-1),trans_40_in,std_40_in,'lineprops','-k','transparent',true);
shadedErrorBar(depth_50_in(1:end-1),trans_50_in,std_50_in,'lineprops','-g','transparent',true);
shadedErrorBar(depth_60_in(1:end-1),trans_60_in,std_60_in,'lineprops','-c','transparent',true);
xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth IN');
legend('20 RPM','30 RPM', '40 RPM','50 RPM','60 RPM','Location','NorthEast');
hold off;


figure(7);
p5 = polyfit(depth_20_out(1:end-1),trans_20_out,2);
f5 = polyval(p5,depth_20_out(1:end-1));
shadedErrorBar(flip(-depth_20_out(1:end-1)),trans_20_out,std_20_out,'lineprops','-b');
xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth 20 RPM OUT');

figure(8);
p6 = polyfit(depth_30_out(1:end-1),trans_30_out,2);
f6 = polyval(p6,depth_30_out(1:end-1));
shadedErrorBar(flip(-depth_30_out(1:end-1)),trans_30_out,std_30_out,'lineprops','-r');
xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth 30 RPM OUT')

figure(9);
p7 = polyfit(depth_40_out(1:end-1),trans_40_out,2);
f7 = polyval(p7,depth_40_out(1:end-1));
```



```
shadedErrorBar(flip(-depth_40_out(1:end-1)),trans_40_out,std_40_out,'lineprops','-k');
xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth 40 RPM OUT')



figure(10);
p8 = polyfit(depth_50_out(1:end-1),trans_50_out,2);
f8 = polyval(p8,depth_50_out(1:end-1));
shadedErrorBar(flip(-depth_50_out(1:end-1)),trans_50_out,std_50_out,'lineprops','-g');
xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth 50 RPM OUT')


figure(11);
p9 = polyfit(depth_60_out(1:end-1),trans_60_out,2);
f9 = polyval(p9,depth_60_out(1:end-1));
shadedErrorBar(flip(-depth_60_out(1:end-1)),trans_60_out,std_60_out,'lineprops','-c');
xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth 60 RPM OUT')


figure(12);
shadedErrorBar(flip(-depth_20_out(1:end-1)),trans_20_out,std_20_out,'lineprops','-b');
hold on;
shadedErrorBar(flip(-depth_30_out(1:end-1)),trans_30_out,std_30_out,'lineprops','-r');
shadedErrorBar(flip(-depth_40_out(1:end-1)),trans_40_out,std_40_out,'lineprops','-k');
shadedErrorBar(flip(-depth_50_out(1:end-1)),trans_50_out,std_50_out,'lineprops','-g');
shadedErrorBar(flip(-depth_60_out(1:end-1)),trans_60_out,std_60_out,'lineprops','-c');
xlabel('Depth, mm');
ylabel(' Velocity, mm/s');
title(' Velocity Vs Depth OUT');
legend('20 RPM','30 RPM', '40 RPM','50 RPM','60 RPM','Location','NorthWest');

data_position;
data_rpm;

speed_20_t1_in = speed_in(rpm_20_t1_in);
speed_20_t2_in = speed_in(rpm_20_t2_in);
speed_20_t3_in = speed_in(rpm_20_t3_in);
speed_20_t4_in = speed_in(rpm_20_t4_in);

speed_20_t1_out = speed_in(rpm_20_t1_out);
speed_20_t2_out = speed_in(rpm_20_t2_out);
speed_20_t3_out = speed_in(rpm_20_t3_out);
speed_20_t4_out = speed_in(rpm_20_t4_out);

speed_30_t1_in = speed_in(rpm_30_t1_in);
speed_30_t2_in = speed_in(rpm_30_t2_in);
speed_30_t3_in = speed_in(rpm_30_t3_in);
speed_30_t4_in = speed_in(rpm_30_t4_in);
speed_30_t5_in = speed_in(rpm_30_t5_in);

speed_30_t1_out = speed_in(rpm_30_t1_out);
speed_30_t2_out = speed_in(rpm_30_t2_out);
speed_30_t3_out = speed_in(rpm_30_t3_out);
speed_30_t4_out = speed_in(rpm_30_t4_out);
```



```
speed_40_t1_in = speed_in(rpm_40_t1_in);
speed_40_t2_in = speed_in(rpm_40_t2_in);
speed_40_t3_in = speed_in(rpm_40_t3_in);
speed_40_t4_in = speed_in(rpm_40_t4_in);
speed_40_t5_in = speed_in(rpm_40_t5_in);

speed_40_t1_out = speed_in(rpm_40_t1_out);
speed_40_t2_out = speed_in(rpm_40_t2_out);
speed_40_t3_out = speed_in(rpm_40_t3_out);
speed_40_t4_out = speed_in(rpm_40_t4_out);
speed_40_t5_out = speed_in(rpm_40_t5_out);

speed_50_t1_in = speed_in(rpm_50_t1_in);
speed_50_t2_in = speed_in(rpm_50_t2_in);
speed_50_t3_in = speed_in(rpm_50_t3_in);
speed_50_t4_in = speed_in(rpm_50_t4_in);
speed_50_t5_in = speed_in(rpm_50_t5_in);

speed_50_t1_out = speed_in(rpm_50_t1_out);
speed_50_t2_out = speed_in(rpm_50_t2_out);
speed_50_t3_out = speed_in(rpm_50_t3_out);
speed_50_t4_out = speed_in(rpm_50_t4_out);
speed_50_t5_out = speed_in(rpm_50_t5_out);

speed_60_t1_in = speed_in(rpm_60_t1_in);
speed_60_t2_in = speed_in(rpm_60_t2_in);
speed_60_t3_in = speed_in(rpm_60_t3_in);
speed_60_t4_in = speed_in(rpm_60_t4_in);
speed_60_t5_in = speed_in(rpm_60_t5_in);

speed_60_t1_out = speed_in(rpm_60_t1_out);
speed_60_t2_out = speed_in(rpm_60_t2_out);
speed_60_t3_out = speed_in(rpm_60_t3_out);
speed_60_t4_out = speed_in(rpm_60_t4_out);
speed_60_t5_out = speed_in(rpm_60_t5_out);
```



```matlab
function a = transvelocity(vel1,time)
[row col] = size(vel1);
a2 = row/240;
a= zeros(round(a2),1);
count = 0;
vel = vel1*1000;

for i= 1:240:(size(vel))
    count = count + 1;
    distance = abs(vel(i+1) -  vel(i));
    time_travelled = abs(time(i+1) - time(i));
    speed = distance/time_travelled;

    a(count) = speed;

end

    a;
end

function speed = speed_in(b)

speed = b(2:2:end);
end
```