

AN ALGORITHM FOR AN IOT PRAWN FEEDER IN CONJUNCTION WITH AN
AQUAPONICS SYSTEM

A Thesis

by

ROWLAND A. RAMOS

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Rainer Fink
Co-Chair of Committee,	Jeonghee Kim
Committee Member,	Mark Benden
Head of Department,	Reza Langari

August 2020

Major Subject: Engineering Technology

Copyright 2020 Rowland Ramos

ABSTRACT

Aquaponics has gained recognition through its food sustainability with different combinations of aquaculture and hydronics. Freshwater prawn (*Macrobrachium Rosenbergii*) can be bought for less than ten cents per prawn and is becoming a popular aquaculture choice for farmers in their aquaponic system. This thesis will demonstrate how developing an algorithm for a freshwater prawn conveyor-based feeder will help farming prawn become a viable option in developing countries and a sustainable food source. Currently, majority of the human population growth is in developing countries where starvation, lack of food sustainability, and lack of resources exists. Here we are today with no devices available specifically for feeding prawn. The current methods are to feed by hand which requires time and effort to measure out the food and dispense it into the habitat. The other choice farmers currently use are deer style feeding system that will either over feed or under feed the prawn which can alter the water characteristics and kill a significant percentage of prawn. We devise an experiment by developing an algorithm with a conveyor feeding system that will dispense food based on the habitat dimensions, maturation of prawns, and the number of prawns. By providing this technology, farmers with any farming background will not have to worry about misfeeding prawn or having to consider all the necessary factors to dispense the correct amount of food. This algorithm has a calibration file that allows it to be used with any conveyor system.

DEDICATION

I dedicate my thesis to God and God alone. Endless love, praise, and thanks to God for allowing me to be in this position. To understanding you God.

ACKNOWLEDGEMENTS

I would like to thank Dr. Fink for being my committee chair and mentor throughout my graduate career. I would like also like to thank my committee members Dr. Kim and Dr. Benden for being a part of my thesis committee.

Thank you to the MSET/ESET faculty for making my experience at Texas A&M University special and full of memories.

Endless thanks and love go out to my family and friends for being there for me every step of the way throughout my academic career. Especially my father and mother.

Lastly, a special thanks goes out to my high school teacher Mrs. Villion for being an inspiration in my life. You are truly special.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Dr. Rainer Fink, Dr. Jeonghee Kim, and Dr. Mark Benden of the Departments of Engineering Technology and Industrial Distribution and Public Health Environmental and Occupation.

Funding Sources

This work was funded by Dr. Rainer Fink through Texas A&M University.

NOMENCLATURE

BP	Bi-Polar
CSS	Cascading Style Sheets
DC	Direct Current
GPIO	General Purpose Input Output
G	Grams
HTML	Hypertext Markup Language
IoT	Internet of Things
JS	JavaScript
MD	Markdown Language
NEMA	National Electrical Manufacturers Association
OS	Operating System
PH	Potential Hydrogen
PCB	Printed Circuit Board
P	Python
ST	Set Time
SPR	Steps Per Revolution
TE	Text Editor
UP	Uni-Polar
USB	Universal Serial Bus

TABLE OF CONTENTS

	Page
ABSTRACT.....	ii
DEDICATION.....	iii
ACKNOWLEDGEMENTS.....	iv
CONTRIBUTORS AND FUNDING SOURCES.....	v
NOMENCLATURE.....	vi
TABLE OF CONTENTS.....	vii
LIST OF FIGURES.....	x
LIST OF TABLES.....	xii
1. INTRODUCTION.....	1
1.1 Problem.....	1
1.2 Proposal.....	3
2. BACKGROUND.....	5
2.1 Macrobrachium Rosenbergi.....	5
2.2 Aquaponic Environment.....	7
3. CONVEYOR SYSTEM.....	9
3.1 Materials.....	9
3.2 Electrical and Mechanical Hardware.....	14
3.3 Connection Setup.....	19
3.4 Construction of Conveyor System.....	24
3.5 Block Diagram of Aquaponic System.....	28
4. SOFTWARE.....	30
4.1 TAMU GitHub.....	30
4.2 Calibration File.....	32
4.3 Feeder Functions.....	35

Main Menu Function.....	35
Habitat Inputs Function.....	35
Habitat Prawns Function.....	36
Net Inputs Function.....	36
Net Prawns Function.....	37
Prawn Weight Function	37
Total Feed Function	38
SPR (Steps Per Revolution) Function.....	38
Cal (Calibration) Ratio Function.....	39
Time Inputs Function.....	39
4.4 Main Project Python File	40
4.5 Test Suite	42
5. RASPBERRY PI CONNECTION METHODS	45
5.1 Connecting Locally.....	45
Peripherals.....	45
Virtual Network Computing (VNC)	45
Putty	46
HTML (Server Based) Research.....	49
5.2 Connecting Remotely	50
Remote Desktop Mobile (App).....	52
HTML (Server Based) Research.....	54
6. RESULTS	55
6.1 Experiment Setup.....	55
6.2 Statistics	55
7. LIMITATIONS AND STRENGTHS	57
7.1 Limitations and Strengths	57
8. CONCLUSION AND FUTURE WORK	59
8.1 Conclusion	59
8.2 Future Work.....	60
REFERENCES	62
APPENDIX A FILENAMES WITH CODE	67
A.1 MainFile.py.....	67
A.2 feederFunctions.py.....	72
A.3 calibrationFile.py	77

A.4 test_Project.py.....	79
A.5 prawnApp.conf.....	80
A.6 prawnApp.wsgi.....	80
A.7 mainProject.py (Server Base)	81
A.8 index.html	82
A.9 position.css.....	82
A.10 prawnScript.js	84

LIST OF FIGURES

	Page
Figure 1. Macrobrachium Rosenbergii (Freshwater Prawn)	5
Figure 2. Prawn Growth Rate	6
Figure 3. Shallow Pool Habitat (30ft x 8ft x 1ft).....	7
Figure 4. Deep Pool Habitat.....	8
Figure 5. Acrylic	9
Figure 6. Wood	10
Figure 7. Nalgene Water Bottle	10
Figure 8. G2 Timing Belt Pulleys	11
Figure 9. Set of Pulleys with Timing Belt	11
Figure 10. Timing Belts	12
Figure 11. Black Linen	12
Figure 12. 10 Gallon Tote.....	13
Figure 13. PVC Pipe	13
Figure 14. Raspberry Pi 3 Model B+	14
Figure 15. NEMA 23-Size Hybrid Stepper Motor.....	15
Figure 16. DRV8825 Stepper Motor Driver Carrier.....	16
Figure 17. HP E3631A External Power Supply.....	16
Figure 18. Schematic for PCB	18
Figure 19. Pin Layout Connections	23
Figure 20. Conveyor System without Tote.....	24
Figure 21. Conveyor System with Tote	25
Figure 22. Part Identification for Pulleys.....	26

Figure 23. Part Identification for Right Side Timing Pulleys 27

Figure 24. Aquaponic Block Diagram 29

Figure 25. All Passed Pytest Output 42

Figure 26. Fail Example Pytest Output 43

Figure 27. VNC Connection 46

Figure 28. Putty Log In via XQuartz 48

Figure 29. Putty Connection (Terminal Window) 48

Figure 30. Web Application Research 49

Figure 31. Port Forward Setup 50

Figure 32. Port Identification 51

Figure 33. Port Forward Finished 51

Figure 34. Log-In Remote Desktop App (iPad) 53

Figure 35. Remotely Connected via iPad 54

LIST OF TABLES

	Page
Table 1. Stocking Densities	6
Table 2. Stepping Format (Microstepping Indexer)	20
Table 3. Stepper Motor Orientation	21
Table 4. Part Identification for Pulleys	26
Table 5. Part Identification for Right Side Timing Pulleys	27
Table 6. Food Dispensed (Actual VS Ideal)	56
Table 7. Percentage Errors and Standard Deviation	56

1. INTRODUCTION

1.1 Problem

As of September 2019, there is a global population of 7.7 billion people and by the end of the century the United Nations predict it will grow to 11.2 billion people [1]. Current trends of sustainable food production has led to serious discussion on the environmental impact of beef and pork production. These discussions have led many to alternative diets including veganism, vegetarianism, and other meat alternatives [2]. However, even these alternative lifestyles come with limitations. Fruits, vegetable, legumes and other vegan sources take time to grow until ready for harvest. Diets based on plants and crops are also potentially unsteady options for a permanent food lifestyle for the mass population, as they are subject to draught, diseases, and limited nutrients. Scientists are currently looking for alternate sources of protein that can sustain a healthy diet [3]. Insects are a potential solution for a protein alternative, in forms of powder, bars, and flour [4]. Unfortunately, it is difficult to convince the general population to consider this as a legitimate protein alternative as it largely differs from the traditional Western diet. Research shows that the growth in population will come from third-world countries [5]. Therefore, it is important to consider a solution from their perspective. There is a lack of affordable and sustainable options currently offered world-wide. Solutions need to accommodate for a lack of resources and potential financial hardship that could limit a third-world country's ability to develop commercial farming practices like the United States.

Aquaponic farming is a growing option for constant, sustainable food production. Prawn is a protein alternative that utilizes this farming method and which warrants further exploration. Prawn are low maintenance, highly sustainable, and easy to raise, requiring little space. Prawn produce ammonium nitrate [6] which feeds the hydroponic system. Large quantities of prawn can easily be produced in small aquaponic farms resulting in an environmentally friendly and sustainable food source. Already efficient farm production can be further improved by utilizing specific algorithms which can automatically adjust to the prawn dietary needs. Algorithms can help the aquaponics community grow, become more efficient and sustainable, and change current farming practices.

Current feeding methods and systems available for prawn in the market have their own challenges. Most affordable automatic feeders are designed for fish and disperse a specific amount per day until the food tank runs out. Traditional feeders are built to attach to popular household fish aquariums. Traditional aquariums' dimensions do not produce enough ammonium nitrate to feed hydroponics in the aquaponics system. High-end timed fish feeders cost thousands of dollars and utilize vibration in order to feed a large amount of fish. Unlike fish, prawn can only eat 5-8% of their body mass [7]. Dispersing too much food into a prawn tank can alter potential hydrogen (pH) levels which could kill the prawn and hydroponics. Traditional feeders are mainly used for commercial fish farming and uses a dial timer to begin and end the feeding process. These timers are manually operated and require a manual reset each time a feeding session is set to occur and cannot provide the need to feed the prawn at least 3 to 4 times automatically throughout the day [8]. Technology today is capable of utilizing software to disperse precise amounts of food

based on the variables that are necessary for prawn in order to grow. It is essential to maintain appropriate balance when utilizing aquaculture in conjunction with hydroponics. Improper balance of the system can destroy the whole aquaponic environment. Currently, there is significant room for growth and development of aquaponic practices. New software solutions and algorithms can change current techniques and develop more efficient and sustainable farming methods.

1.2 Proposal

This project utilizes an inexpensive conveyor system built to distribute food into the tank of prawn. The system consists of a pulley system controlled by a stepper motor. To control the stepper motor, it is connected to a Raspberry Pi, a small single board computer (SBC) via its general-purpose input output (GPIO) pins. The conveyor belt needs to produce enough friction and withstand high levels of humidity. This conveyor system utilizes inexpensive materials, so when components wear out with use or break, they can be easily replaced. In another section of this thesis, a list of materials and the design used are available for reproduction. The algorithm is independent of any conveyor system used. The solution that is developed and implemented on the Raspberry Pi is an algorithm to determine the precise amount of food that should be dispersed. The algorithm is based on the dimensions of the habitat, number and maturation of the prawn. It will then take this information and decide how much to food it needs to disperse that day so that every single prawn can eat according to their dietary needs. Freshwater prawn is a territorial species and will kill if they are overpopulated or encroached upon. The algorithm will be able to take this into account and ensure that a warning is provided that

only “n” number of prawns should be allotted based on the dimensions of the habitat. As the days go by, the prawn will go grow and the food dispensed will increase in correlation with the prawn’s mass multiplied by the number of prawns allotted. This needs to be done over a 170-day span to reach an individual prawn’s peak mass which is between sixty-five and seventy-three grams and a length of six to seven inches [9]. The current time and date are also utilized so the system can adapt to any scenario with any combination of maturity level and quantity of prawn. This software will then determine the correct number of days left for the prawn to mature, and dispense the food accordingly.

2. BACKGROUND

2.1 *Macrobrachium Rosenbergii*

Macrobrachium Rosenbergii is the scientific name for freshwater prawn and there are over 200 species throughout the world [9]. A reprinted picture of freshwater prawn can be seen from figure 1. Prawn are omnivorous and coprophagous which means that they can feed off of both plant or animal or feces respectively [7]. Prawn are also a territorial species and require different stocking densities depending on the maturation period [10]. Figure 2 showcases the growing rates for prawn per pound that were adapted from [10]. The characteristics from table 1 are adapted from [10] and shows the stocking densities for prawn based on the prawn's maturation period. A fully-grown prawn at 160-170 days can weigh 45 grams which is about 10 prawn per pound and have a length of 7 inches [10]. Prawn are also a tropical species that require water temperatures to be between 78° and 84° Fahrenheit (25° C and 29° C) [10].



Figure 1. *Macrobrachium Rosenbergii* (Freshwater Prawn)

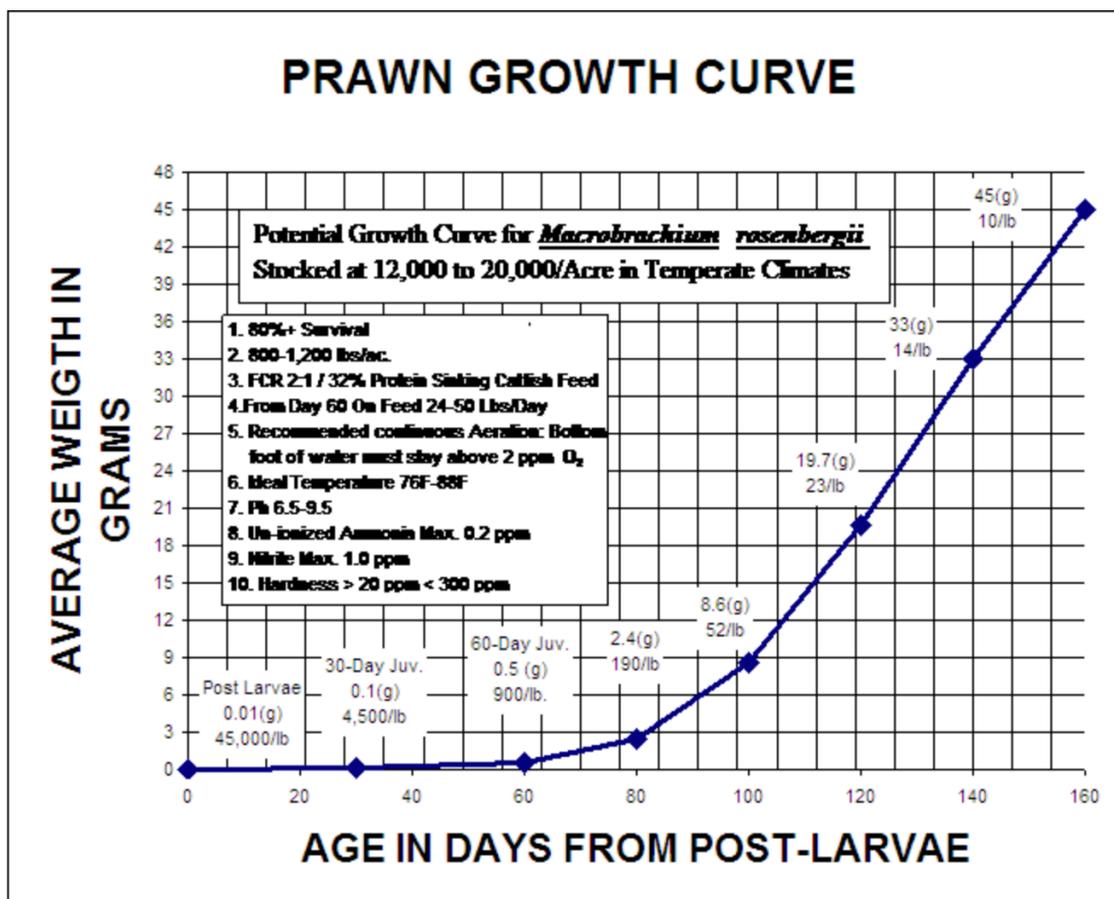


Figure 2. Prawn Growth Rate. Reprinted with Permission from [10]

Table 1. Stocking Densities. Reprinted with Permission from [10]

Maturation Period (Days)	Number of Animals per ft ²
45	40
60	20
60-90	2
> 90	1

2.2 Aquaponic Environment

Figure 3 shows a shallow pool habitat for prawn that is about one foot deep. The water from the prawn pool slowly gets distributed into another pool of water that contains the hydroponics that feed off of the nitrate and nitrite that the prawn produce. The water then gets pumped backed into the prawn pool to create an eco-friendly environment. Having a shallow habitat allows the farmer to see the progress of the prawn without having to take the prawn out of the pool. Another plus about this pool is the cost of water is lower and there is less maintenance that is required. A downfall for having a shallow pool is the limited number of prawns that can be grown due to the dimensions.



Figure 3. Shallow Pool Habitat (30ft x 8ft x 1ft)

The deep pool habitat shown in figure 4, has the same eco-friendly system with the hydroponics but its depth is about four feet. A unique feature that can be implemented with this pool is the addition of nets. The nets allow the allotted number of prawns to live in the habitat to increase because they provide more surface area that the prawn can live on. Having this pool makes the cost of water more expensive and the prawn on the surface of the water are only visible, making it difficult to see the progress of prawn that are living on the bottom of the pool.



Figure 4. Deep Pool Habitat

3. CONVEYOR SYSTEM

3.1 Materials

Most small scale aquaponic environments are not temperature controlled and result in high temperatures and humidity. Durability is an important consideration for the design of the conveyer system. Shown in figure 5 is Acrylic, also known as plexiglass will not warp if wet and is an appropriate material for the demanding environment. Acrylic is made of petroleum thermoplastic and while it is a type of plastic not all plastics are acrylic [11]. The side wall dimensions are 9 inches by 24 inches.



Figure 5. Acrylic

The acrylic walls are fixed together by five pieces of wood shown in figure 6. Wood is an easy material to drill screws into and strong enough to withstand at least ten gallons of prawn food. Other materials were considered but unavailable secondary to the COVID-19 pandemic. The dimensions of the wood are 1.5 inches by 12 inches.



Figure 6. Wood

The conveyer belt requires rollers in order for the belt to freely rotate. Cylindrical pieces of wood were initially utilized but failed secondary to improper technique. Without a drill press, precision was lacking. Equipment limitations required adaptability. Instead of cylindrical wood rollers, Nalgene® water bottles were utilized shown in figure 7. They are both durable and symmetrical. Nalgene water bottles are created with extremely tough plastic and is impact resistant. The top of the water bottle has a large enough diameter so that a pulley can be mounted on the lid.



Figure 7. Nalgene Water Bottle

To keep the water bottles from shifting left and right, eight timing belt pulleys were fixed to each side of the water bottle shown in figure 8. An additional pulley was placed on the outside of the acrylic wall to keep the shaft from moving side to side.



Figure 8. G2 Timing Belt Pulleys

Another major part implemented is a pulley combination that would fit the shaft of the stepper motor which was 6.35mm and larger pulley that could turn the water bottle. The pulleys that are used in this system were designed for 3-D printing but worked perfectly for this application. In Figure 9 show the set of the pulleys and the timing belt it came with.

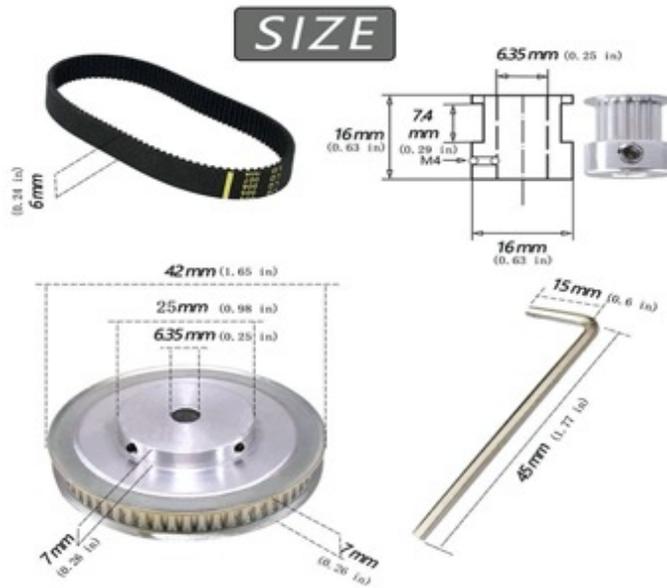


Figure 9. Set of Pulleys with Timing Belt

During construction, it was unknown what length was needed to connect both pulleys together so a variety was bought to determine the length as shown in figure 10. The length that worked with the spacing of the stepper motor and the large pulley was a 5 ½ inches timing belt with a width of 6mm.

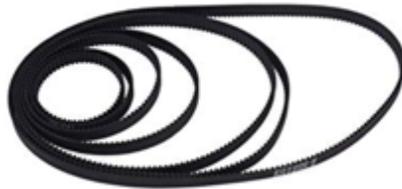


Figure 10. Timing Belts

For the conveyor belt, black linen was used with Velcro to complete the loop as shown in figure 11. The linen is strong enough to carry the weight of the food and undergo humid conditions. It is extremely inexpensive to replace and can be easily washed if it gets dirty over time.



Figure 11. Black Linen

Next, to store the food securely, a ten-gallon tote was placed on top of the conveyor system and PVC pipe hovered over the conveyor belt where the food is going to be dispensed as shown in figure 12. A ten-gallon tote is a sufficient volume to test with and

is a perfect fit to place on top of the wood beams that were one foot in length. The tote has a lockable lid and is resistant to outside bugs and animals.



Figure 12. 10 Gallon Tote

Lastly, a hole inside was created at the bottom of the tote and polyvinyl chloride (PVC) pipe was placed through the hole to provide a way for the food to travel from the tote to the conveyor belt. PVC as shown in figure 13, is strong and is able withstand high pressure and humidity. It is widely used in plumbing can handle high pressures of water.



Figure 13. PVC Pipe

3.2 Electrical and Mechanical Hardware

The brains behind this system is a Raspberry Pi 3 Model B+ that has a 1.4GHz 64 bit quad-core processor and has 1GB of random-access memory (RAM) shown in figure 14. This small single board computer (SBC) enables the algorithm be developed in Python instead of C programming language. Python is a well-known and utilized program with an abundance of documentation for programming language libraries. The processing power will be able to read and write data to the internet with low latency. Low latency enables precise calibration when creating the algorithm. The Raspberry Pi will easily handle the internet of things (IoT) system. Lastly, this SBC contains a microchip and supports Wi-Fi capabilities.



Figure 14. Raspberry Pi 3 Model B+

There are two primary kinds of National Electrical Manufacturers Association (NEMA) stepper motors, bipolar and unipolar. Bipolar stepper motors provide more torque because their coils are twice as large, but provide less control during programming- Fewer phases inside the motor allow less degrees of freedom. Unipolar stepper motors

have two windings per phase providing better control of the motor with improved precision. This results in more control of the motor, but creates less torque. The motor for the conveyor system is a NEMA 23 Stepper motor with 200 steps per revolution (steps/rev), 7.4 Volts, and 1 A per phase hybrid (1A/phase) stepper motor shown in figure 15. Having 200 steps/rev enables control of the motor as low of a 1.8° step angle. Lastly, a hybrid motor can be configured to be bipolar or unipolar setting. The bipolar setting was implemented and provided enough torque and control for the food to be dispersed accurately.



Figure 15. NEMA 23-Size Hybrid Stepper Motor

The stepper motor driver in the conveyor system gives adequate power and directional control. The stepper motor driver connects directly to the Raspberry Pi for inputs and outputs when programming in Python. The motor driver also connects to the stepper motor to give power and direction. The motor driver has a pin for voltage and two H-bridges in the DRV8825 shown in figure 16. An H-bridge controls the direction of current with metal-oxide-semiconductor field-effect-transistors (MOSFETS). When sending current to both phases, the H-bridge will allow current to flow in one direction.

The opposite logic it will turn off two MOSFETS and turn on the other two allowing the current to flow in the opposite direction. This motor driver will not require any external devices because it has an internal voltage regulator and a potentiometer to control the current.

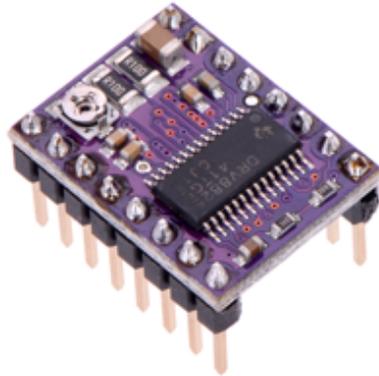


Figure 16. DRV8825 Stepper Motor Driver Carrier

To power the motor of the conveyor system, an external power supply is required to provide sufficient voltage and current. A Hewlett Packard (HP) E3631A was used in the conveyor system as shown in figure 17.



Figure 17. HP E3631A External Power Supply

Lastly, a schematic was designed in order to create a printed circuit board (PCB) in the future that could be used for the Raspberry Pi, Stepper Motor Driver, Power, and the NEMA 23 Stepper Motor shown in figure 18. This eliminates any connecting wires from and to each device and further prevents any corrosion in the aquaponics environment. It also functions as one unit and could be stored easily onto the conveyor system into a water container. The PCB consists of three headers, one four-pin female connector, one two-pin female connector, and a capacitor that is used for the two-pin connector. One of the three headers is a 2 x 20 pin female header for the Raspberry Pi. From this header, it will go to one of the 1 x 8 pin female header for the stepper motor driver. These connections made will allow it strictly to communicate from the Raspberry Pi to the motor driver. The last header is also a 1 x 8 pin female header used for the stepper motor. The first two pins from the header go through the 100uF 16V capacitor to the 2-pin female connector that is used for providing voltage and current to the stepper motor. The current is placed in parallel to prevent any voltage spikes from the power supply. The next four pins from the header runs to the 4-pin female connector that will be used to plug the stepper motor wires into. This will allow the motor to rotate by connecting the stepper motor driver. This PCB is a simple design because the motor driver has all the necessary components integrated into it such as a voltage regulator and a potentiometer. It is made to plug and play the primary devices without any need of additional wires or components to complete the electrical hardware necessary for the conveyor system. Figure 14 shows the schematic created in EAGLE which is PCB software in conjunction with SnapEDA. SnapEDA is a search engine for electrical components that can be integrated into the

schematic. This is search engine provides the footprint that are needed to allow for the connections of pins, connectors, and devices.

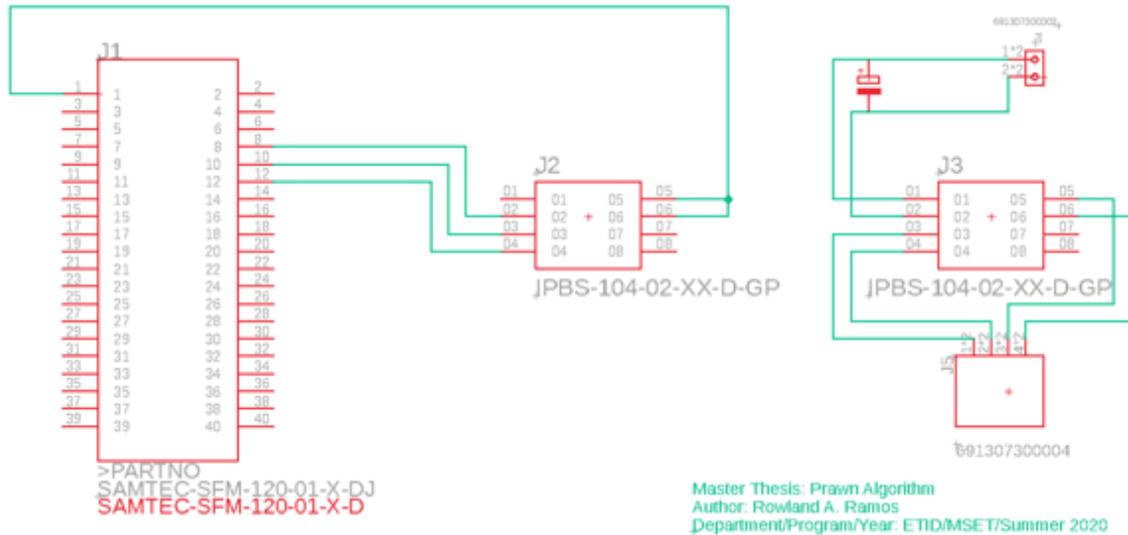


Figure 18. Schematic for PCB

3.3 Connection Setup

The connection will start with the Raspberry Pi to the stepper motor driver. There is a 3.3 Volts that is going to be supplied from the Raspberry Pi to the RST and SLP pins on the stepper motor driver in order to shut off the pins because they are naturally active low inputs. The RST pin is called the reset pin and resets the internal logic of the DRV8825 along with resetting the step table to the home position [12]. If this pin is not pulled high the motor will not rotate because it will constantly be resetting the logic not allowing the STEP pin to activate. The SLP pin is the sleep pin that puts the device in a low power mode or which Texas Instruments calls a sleep mode [12]. If this pin is active it will disable the internal voltage regulator, the internal clocks will become inactive, and the two H-bridges will be disabled not allowing current will to flow through the phases of the motor [12].

Next the general-purpose input output (GPIO) pins 14, 15, and 18 from the Raspberry Pi will connect to the M0, M1, and M2 pins on the motor driver respectively. These GPIO pins will be setup as outputs because they will send a digital high or low signal to the M pins on the stepper motor driver. The M0 pin on the motor driver stands for mode and is used for microstepping feature for the motor. In order to use this feature all mode pins will be utilized which are M0, M1, and M2. Microstepping is used to control the resolution (stability and smoothness) of the motor but in doing so the motor will lose torque and speed the larger the micro step. For example, in order to achieve a full step requirement, we look at table 2 of the stepper motor. There is a 1.8° step angle for this motor so we can calculate the steps for one revolution [12].

$$\frac{1 \text{ Revolution}}{\text{Step Angle}} = \frac{360^\circ}{1.8^\circ} = 200 \frac{\text{Steps}}{\text{Rev}}$$

Once calculated, the amount of time needed to complete one revolution by dividing one second by the number of steps per revolution which is 0.005 seconds or 5 milliseconds (ms).

Table 2. Stepping Format (Microstepping Indexer)

Mode2	Mode1	Mode0	Step Mode
0	0	0	Full Step (2-Phase Excitation) w/ 71% Current
0	0	1	½ Step (1-2 Phase Excitation)
0	1	0	¼ Step (1-4 Phase Excitation)
0	1	1	8 Microsteps/Step
1	0	0	16 Microsteps/Step
1	0	1	32 Microsteps/Step
1	1	0	32 Microsteps/Step
1	1	1	32 Microsteps/Step

The next pin connection from the Raspberry Pi is GPIO 21 and it is used to control the STEP pin on the motor driver. The STEP pin is going to receive digital high or low inputs and will allow the STEP interface to advance the indexer through the transition from low to high input [12]. In other words, the indexer travels to the next state during the rising edge of the STEP pin [12].

The last GPIO pin that is used from the Raspberry Pi is GPIO 20 that is connected to the DIR pin on the stepper motor driver. DIR is short for direction and it is not necessary if the motor does not need to go counter counterclockwise (CCW). Setting up GPIO 20 as an output direction will be controlled by sending a digital low for counterclockwise or digital high for clockwise (CW) as shown in table 3. The final pin from the Raspberry Pi is the ground pin to ground pin on the stepper motor.

Table 3. Stepper Motor Orientation

Digital Signal to DIR Pin	Direction
0	Counterclockwise
1	Clockwise

Before moving on to the power supply connections it is important to note that the stepper motor can only handle 1A/phase [13]. The stepper motor driver has an internal potentiometer that can be adjusted to fit the needs of the stepper motor specifications. If this is not done before turning on the power supply there is a good chance that the current will burn up the internals of the stepper motor. Another safety measure that was applied was placing a 100 μ F 16V rated capacitor. The capacitor protects the stepper motor from voltage spikes that the power supply can accidentally have. In order to set the potentiometer at the correct resistance the following formula is used [13].

$$\text{Current Limit} = V_{\text{ref}} * 2$$

Since the stepper motor can handle only 1A/phase we then can do a simple calculation that V_{ref} should be 0.5 Volts and is measured with a multimeter at the reference point on the motor driver.

After setting the correct voltage reference, power and ground needs to be connected from the power supply to the motor driver. Providing 1 A at 7.4 Volts from the power supply for each phase to the motor driver will power the motor and give it the necessary current in order to function [12].

Lastly, after some testing, the stepper motor was configured to a bipolar configuration and was optimal for the setup of the conveyor system. In Figure 19 each phase has a specific color that indicates which wires belong to what phase. In order to run the NEMA 23 stepper motor in the bipolar configuration the middle wires for each phase (yellow and white) are ignored because they are used for a ground. One of the phases from the stepper motor is going to pins B2 and B1 on the motor driver because those specific pins go through one of the H-bridges that controls the direction and speed of the motor. The other phase is going to pins A2 and A1 which has the second H-bridge. An important thing to note is the orientation of the wires. If one of the phase wires were swapped on the motor driver pins the motor would not rotate because the currents from each H-bridge would cancel each other out. The connections need to be setup so that current is flowing in one direction through the phases. It can be seen more clearly looking at the DRV8825 datasheet on page eleven [12].

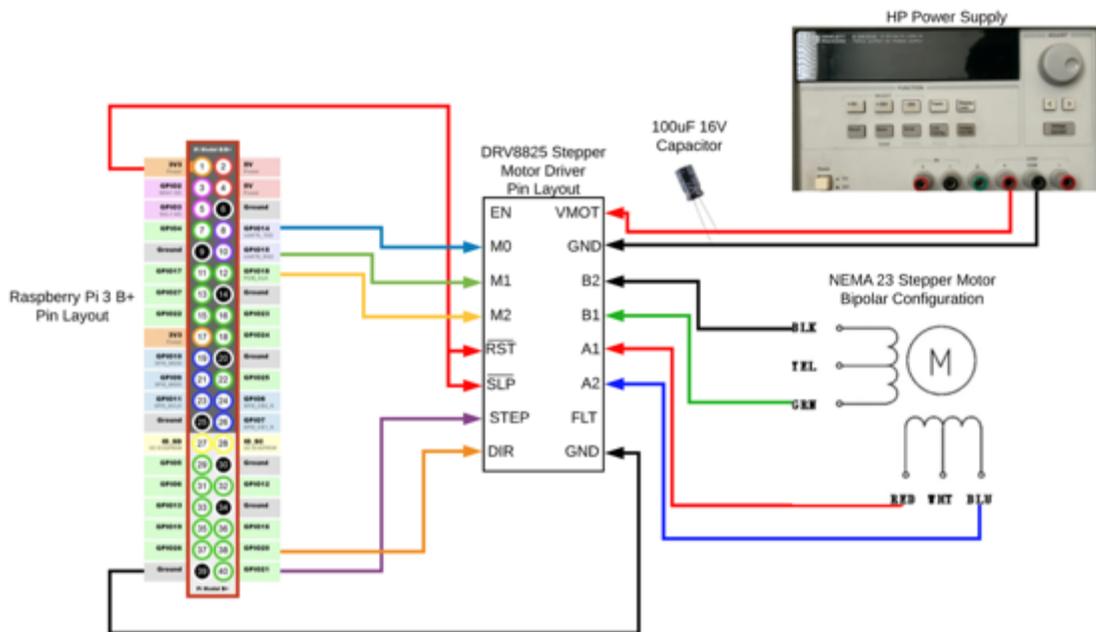


Figure 19. Pin Layout Connections

3.4 Construction of Conveyor System

The conveyor system was designed to be as small as possible so it can attach to the habitat of prawns easily. It is important to note that the prawn algorithm will work with any existing conveyor system. To convert an existing system, it just needs to run the calibration file that will be explained later. The dimensions of the conveyor system without the tote is 24 inches (length) x 12.5 inches (width) x 9 inches (height) as shown in figure 20. With the stepper motor and tote attached the dimensions are 24 inches (length) x 14.75 inches (width) x 21 inches (height). In figure 20 and 21 show the setup and finished construction of the conveyor system with and without the tote respectively. The tote has a volume of ten gallons and it is used to store and place food on the conveyor belt.

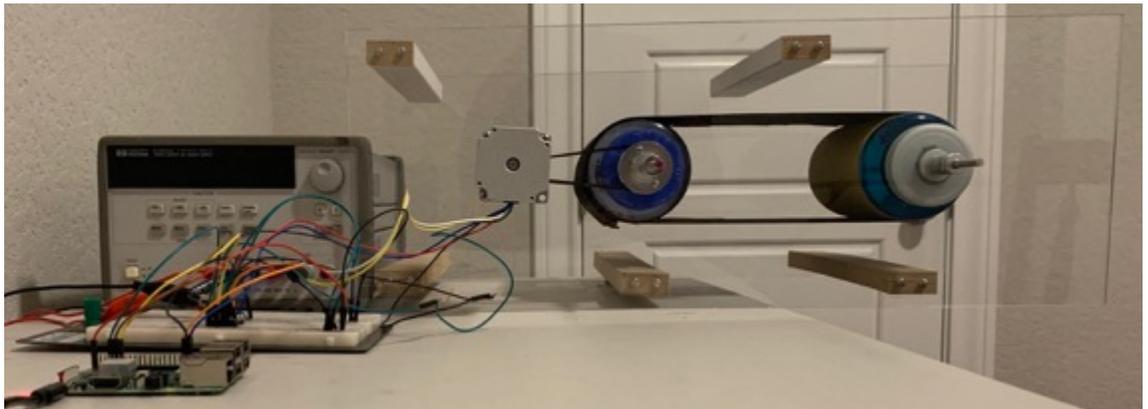


Figure 20. Conveyor System without Tote

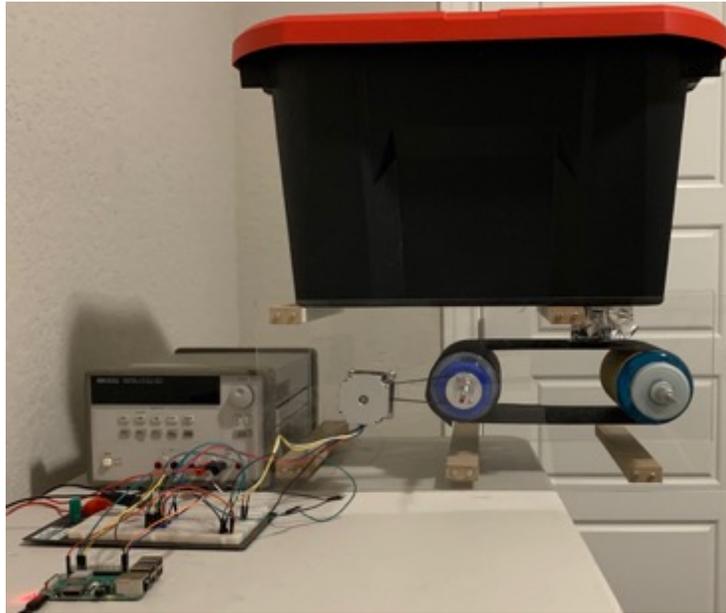


Figure 21. Conveyor System with Tote

Next, figure 22 shows where the large pulleys were placed and table to clearly identify what parts are placed where. In the same figure, the number one in the highlighted blue square, is the NEMA 23 stepper motor and it is mounted into the acrylic with four screws and four nuts. The motor has a unique D shaft and has a diameter of 6.35mm (1/4 inch) that goes into the plexiglass and out where it connects to the G2 Timing Pulley represented by a number two in the red circle. The bore of the timing pulley is also a quarter inch and has two set screws that are tightened to the shaft of the stepper motor so that the pulley is secure. The number three in the highlighted yellow oval represents the two-timing pulleys. The outer pulley is used to stop the shaft that goes straight through the whole system from shifting right. The inner timing pulley is used to align the water bottles so the conveyor belt rotates straight. Lastly, the number four in the highlighted

green oval is the larger pulley that is 1.65 inches in diameter and is screwed on the center of the cap of the water bottle. Table 4 reflects figure 22 as a legend.

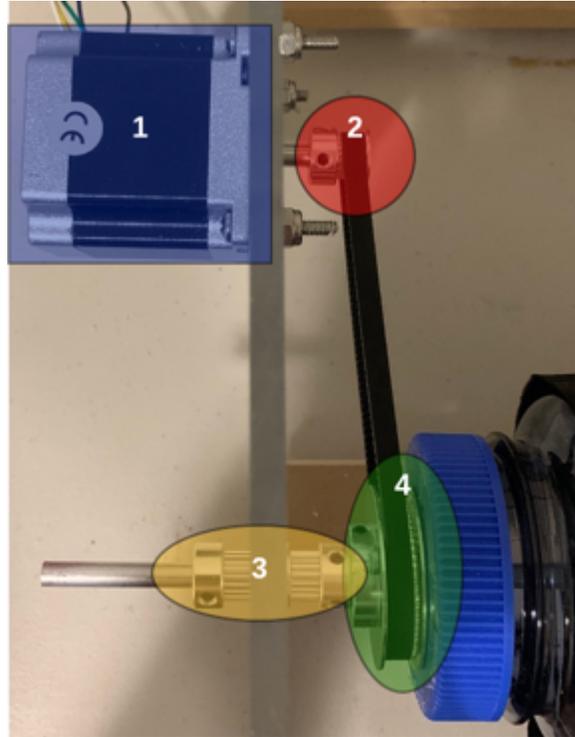


Figure 22. Part Identification for Pulleys

Table 4. Part Identification for Pulleys

Highlighted Color	Number Identity	Part	Purpose
Blue	1	Stepper Motor	Rotating pulleys for conveyor belt
Red	2	G2 Timing Pulley	Connecting Timing Belt
Yellow	3	G2 Timing Pulley	Prevent right shift/alignment
Green	4	Large Pulley	Connecting timing belt

Figure 23 shows a better angle where the other two-timing pulley are located. It also shows how the food will be dispensed once on the conveyor belt. One of them is located at the bottom end of the water bottle and is numbered one in the highlighted blue. This timing pulley maintains the alignment of the conveyor belt by keeping the water bottle stationary. The other timing pulley is on the outside of the acrylic wall and is represented by the number 2 in the highlighted in a red circle. This specific timing pulley keeps the straight through shaft from shifting left. Table 5 reflects figure 23 as a legend.

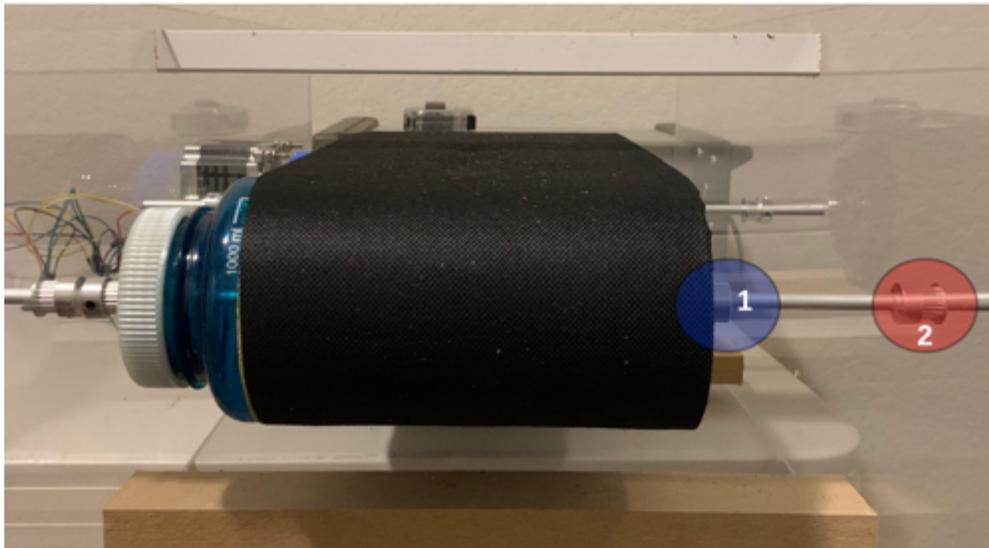


Figure 23. Part Identification for Right Side Timing Pulleys

Table 5. Part Identification for Right Side Timing Pulleys

Highlighted Color	Number Identity	Part	Purpose
Blue	1	G2 Timing Pulley	Alignment
Red	2	G2 Timing Pulley	Prevent left shift

3.5 Block Diagram of Aquaponic System

This block diagram in Figure 24 is an overall representation of the aquaponic environment that the conveyor system is going to be installed in conjunction with. The conveyor system will be mounted onto the prawn pool so that it can dispense the appropriate amount of food. To start the algorithm the user will have to connect to the Raspberry Pi locally or remotely and put in all of the necessary information. Another smart IoT device is the sensors that read oxygen saturation (SpO_2), Turbidity, pH levels, temperature, and water flow to check for leaks or evaporation levels. The water circulates from the prawn habitat which contains nitrite (NO_2^-) and nitrate (NO_3^-) that prawn produce to the hydroponics which feed off of it. The water gets filtered by the hydroponics and gets pumped back into the prawn habitat with the filtered water finishing the circulation. All of this data from the sensors and Raspberry Pi gets saved in a text file so it can be studied.

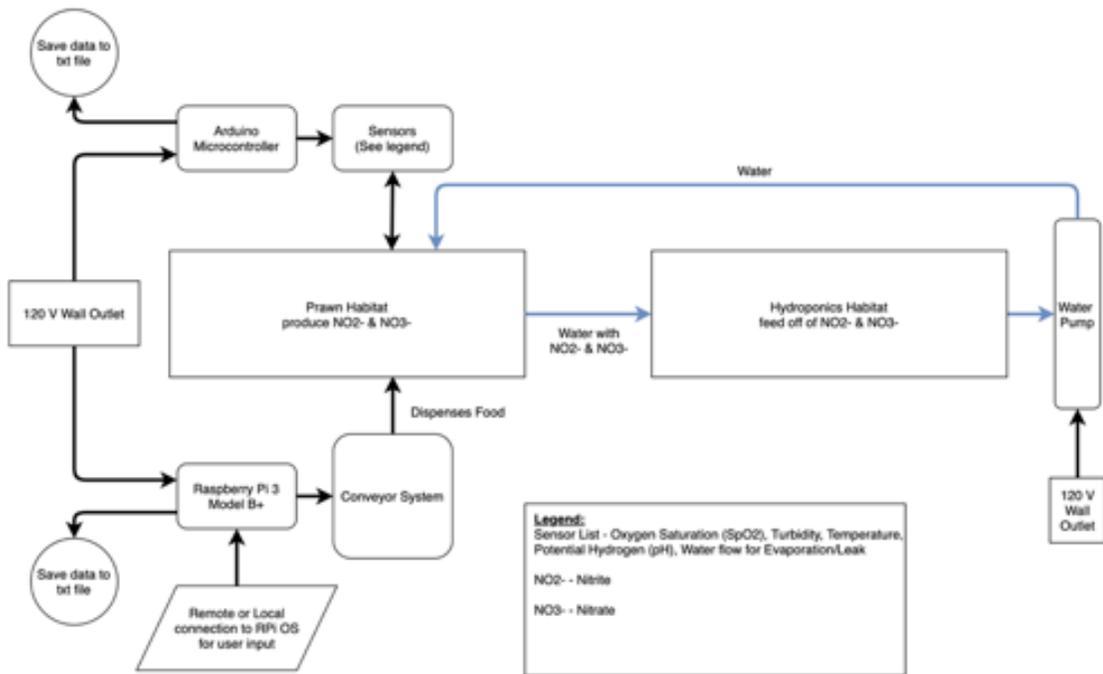


Figure 24. Aquaponic Block Diagram

4. SOFTWARE

4.1 TAMU GitHub

Before starting on the software for this algorithm, a remote repository was created on Texas A&M University's (TAMU) GitHub. GitHub is version control that allows developers, researchers, and people doing software projects to safely upload their code online. This enabled a local repository on a personal computer. A local repository is a local directory (folder) that gives the ease to press a single button to push the new file changes to the remote repository. This also gives the opportunity for people that are interested in research to look at saved files, and determine if existing work would benefit their own research. Another great feature, if this was a project involved multiple people, each member on the team could clone the same local repository onto their personal computers and work on different sections of the software. GitHub is known to all programming languages and has a vast library of projects. Creating this repository enables a secure code in case any happens to the local computer on which the algorithm is developed. One of the most important details in the GitHub community, is creating a read me and manual file for a project. The read me file is formatted in markdown (MD) file that gives a brief introduction of what the project is about and what dependencies it entails. The dependency is a description of what programming language the project is developed in and what libraries you need to include.

If the user decided that this project can benefit them, they can read an in-depth summary. It will contain various topics that include the Introduction, Using the Application, Terminal Output Example, Dependencies, and Test Suite. The title of the

manual has the author's name along with the school or organization, where the author is from and the latest software version it has. Since this is the first version, it is numbered as 1.0.0 and once updated with a correction or optimized version, the newest version will be available as well.

The introduction of the manual gives a summary of what the software does and what outputs it will provide. Next, using this application section will give the user an in-depth explanation of what to expect when running the main Python file. The better this section is explained, the less reverse engineering the user has to do when looking at the Python files. The user will know what to expect instead of having to understand each line of code. After the user is done reading an output example that will be displayed from the terminal. The example in manual shows the output for one whole day that the software was ran. The next topic is the dependencies which tells the user what internal libraries they will need to import and what external library they will need to download from the terminal. It also shows the command on how to install the external library. Finally, the last section is the test suite and it explains what it is used for and how to tun it.

4.2 Calibration File

The calibration file is a function called `motor` that takes in the steps per revolution (SPR) as an argument. In order to run this function properly there are three libraries that need to be imported which are `from time import sleep`, `import RPi.GPIO as GPIO`, and `from time import perf_counter`. Inside the function the first thing that is coded is to turn off the Raspberry Pi's GPIO warnings. The warning is just to provide a caution to not use a single port for two functionalities in your system. Code needs to be implemented to setup the direction interface to GPIO 20 on the Raspberry Pi by assigning it an integer value of twenty. The same code is written when setting up the STEP port to GPIO 21. The next variables that were set up is the clockwise (CW) and the counterclockwise (CCW) values. CW will take in an integer value of one and CCW will take in an integer value of zero. After setting up these variables I need to set them as inputs or outputs depending on their function. Before setting the variables as inputs or outputs the `setmode` needed to be configured within the code. `setmode` allows the header on the Raspberry Pi to be called from the GPIO number or the pin number. For example, for GPIO21 which is pin 40 on the Raspberry Pi J8 header, in code it can be coded in Python "21" to program to that specific interface if Broadcom SOC Channel (BCM) is used. If Board `setmode` was used, the code has to refer to the pin which is 40 [14]. After the `setmode` was programmed the direction and the step variables were programmed as outputs. The direction was coded to rotate CCW based on the construction of the conveyor system.

After setting up the step and direction the mode GPIO pins were coded as outputs which are 14, 15, and 18. These mode pins control the microstepping which allows for a

better resolution. Next, a dictionary was created to test what stepping method worked best for the feeder. The dictionary reflects what table 1 shows for the microstepping index. After some testing different microstepping modes, half stepping was applied for the conveyor system because the speed and resolution was a perfect fit. Since this stepping method was chosen it now takes the motor double the steps per to complete one revolution. This motor at full step has 200 steps per revolution so the motor now needed to have 400 steps to complete one revolution. The time it takes to make one revolution at full step now takes half of the time which is 5ms divided two which is 2.5ms to complete one revolution. After these calculations, a time stamp was coded and placed in a variable called “initialTime” so the user can tell how long it will take for the motor to start. Next is the main part of this function which is rotating the motor. This is done by creating a for loop to run as long as the range of the step count that was doubled because of half stepping. Inside the loop the step pin is called as a digital high and as mentioned before the indexer only transitions on the rising edge of the digital signal. The sleep function is used by setting it to the value calculated which was 2.5ms. After a digital low signal is then sent to the step pin, and finally the sleep function is called once again. This will continue to run until the for loop is finished with the step count (motor finished rotating). Once finished, I took another time stamp and named it “finalTime” and the elapsed time can is now attainable by subtracting the final time by the initial time. To finish the writing the motor function, another function in the GPIO library called “cleanup” is called. This clears the logic from all ports on the Raspberry Pi so next time the ports are in use it prevents it from having left over instruction.

The conveyor belt in the system is thirteen inches long and the food is dropped off four inches from the edge where it is dispensed into the tank. In order to calibrate correctly a scale is needed that can measure in grams to weigh the food when the motor function is executed. From the prawn feeder that was constructed the food weighing results were measured at an average of 17 grams \pm 5 grams per one hundred SPR (100 SPR = 17 grams). This ratio is important when developing the algorithm in the main project Python file. Keep in mind that this ratio is based on this specific conveyor system and other users that go by these numbers will have different results when calibrating on their own prawn feeder. It is recommended that the user runs the calibration at least ten times and takes the average weight of food for the total times and make that part of the ratio.

4.3 Feeder Functions

Main Menu Function

This function was created to allow for easy navigation when the main python file is executed. A menu is displayed on what actions can be carried which is a total of eight. The first six options are required to be completed before running the algorithm because the algorithm is dependent on all of the user input data. Once all of the information is completed the algorithm will calculate everything in order to successfully run and until the harvest day arrives. A great feature about creating a menu is if the user inputs the incorrect information or accidentally hits the wrong button, the user can re-run the option as many times as needed to put in the correct information.

Habitat Inputs Function

The Habitat Inputs function was created to gather the dimensions of the habitat which is the length, width, and height. There is a limitation where any possible dimension cannot exceed over fifty feet. The reason for this is because this algorithm is created for small scale production aquaponic systems. Also, the function takes in whole number answers and will give instruction to round up or down based on the user measurements. Dealing with whole numbers is important because it is all going to relate on how the SPR will be calculated which will then dispense the correct amount of food. After the function gathers all of the three dimensions it will store then inside of an array (list) and return then to the main Python file so that the numbers can be utilized. This function also has a check safe where if the user inputs any input other than an integer it will provide a message saying that it is an invalid input.

Habitat Prawns Function

This function calculates how many prawns are allotted in your habitat based on the dimensions the user input. The function takes in three arguments which are the length, width, and height. Prawns need at minimum one square foot of room to live in because they are territorial crustaceans. That is a primary reason why the dimensions are measured in feet. The function will first calculate the how many prawns the starboard and portside walls can hold. Then the function will calculate how many prawns can live on the deck of the habitat. Next, it will calculate how many prawns can live on the fore and aft wall. Lastly, this function sums up the total of prawns from each wall and the deck and return the value to the main Python file so it can be used for to calculate the right value for the SPR.

Net Inputs Function

There are practices where the habitat has multiple nets inside so that the surface area of the nets can provide space so more prawn can live within the habitat. This function takes in no arguments, but takes in user input. The same concept from the habitat input function, any dimension of the net cannot be above fifty feet and have to be a whole number. The function will ask for the length, width, and the number of nets that are going to be used in the habitat. These values are an important to find the total amount of prawn that can live on the surface area of the nets. Once all values are gathered it stores them into an array and returns the array the main Python function to use used. This function also has preventative measures where it will display a message invalid input if a character,

group of characters, or a floating point (ex: 5.93) is entered. If there are no nets the user can simply place zeros for all three inputs.

Net Prawns Function

The Net Prawns function takes in three arguments which are the length, width, and the number of nets. Once the values enter the function it will then calculate the total number of prawns that can live on the surface area of the total nets. Once the total prawns are calculated it returns this value to the main Python file to be used in order to calculate the SPR which is used to dispense the correct amount of food.

Prawn Weight Function

Prawn only eat five to eight percent of their body mass so it is important to know what the individual prawn weighs and what is the maturation period of the prawn (How old an individual is) when trying to dispense the correct amount of prawn feed. When prawn nurseries sell prawn, they usually sell them around twenty to forty days old which a prawn on average will weigh 0.15 grams. With this information, the function takes in one argument which is called option. The user will be displayed this information and will be given a choice of two options. One option, which is the recommended option, is typing in “normal” which puts the maturation period for every prawn at thirty days and the weight of 0.15 grams. If the user knows they bought the prawn at ninety days and each prawn weight is seventy grams they can type in “other.” By doing this the program will ask for user input for these two characteristics. Once the user inputs this information the function will return the maturation period and prawn weight to the main Python function to be integrated for the SPR.

Total Feed Function

The total feed function takes in two arguments which are the total number of prawns and the weight of the individual prawn. It will then calculate the total amount of prawn feed at eight percent of their body mass. I decided not to go with five percent just because. There wasn't no deciding factor that was led to this, just a decisive decision to see if this percent will yield better results when feeding the prawn.

SPR (Steps Per Revolution) Function

The SPR function is the key to getting the correct steps for the motor to dispense the right amount of food for the prawn. The function takes in three arguments which is "spr", "foodDispensed", and "totalFeed". This is where the calibration comes into play because the lower case spr is for the value you used when measuring the food dispensed during the calibration. For instance, when calibrated the spr to food ratio in section 3.2 the ratio is 100 spr to 17 grams of food. The important part is the weigh the food after each motor run and calculate the average grams of food dispensed and keep the spr static throughout the calibration. This is the ratio that is used for this function and it will be different for every conveyor system this is tried on. Once spr and food dispensed are accounted for the last argument that is needed is the total feed which you received from the total feed function. The SPR function will then calculate the new SPR value that will dispense the total amount of prawn feed needed for the habitat.

Cal (Calibration) Ratio Function

The calibration ratio function takes user input for the steps per revolution and the average food dispensed at that specific SPR. The function then returns those values so it can be used to calculate the SPR that is going to be used for the total amount of prawn in the habitat.

Time Inputs Function

The time inputs function takes in one argument which is the number of times the user wants to feed the prawn in one day. This value is needed because based off the answer, which is going to be three or four, will then ask the user for times throughout the day they would like to feed the prawn. The function displays a message indicating that the user should input the times in military format. For example, if one of the times is going to be 06:30 PM the user should type in 1830. Another message that it displays is the user should input the earliest time to the latest time in the day. If user fails to do so, the function corrects the error by putting in the correct time order. Lastly, the function returns all of the user time inputs to the main Python to be used for the execution of dispensed food at those specific times.

4.4 Main Project Python File

In order to run the algorithm, you must run the “mainProject.py” in terminal. This file is dependent on the “calibrationFunction.py” and “feederFunctions.py” files or else it will not work. All three files must be located in the same directory (folder) so that the files can be imported into the main project file. The main project is dependent on libraries that include math, time, and from datetime import date, datetime, and timedelta. To import other files its similar to importing libraries. For instance, to import all of the files into the main you write “from feederFunctions import *”. The asterisk in the line means that you are importing all of the functions within the file so it alleviates having to import them one by one. After the libraries and files have been imported, the functions are called that were created for the algorithm. In each file the functions are placed in specific order so it can match the flow when called in the main Python file. The first function called is the habitat inputs to get dimension values and calculate the total number of prawns that can be allowed in the habitat alone. Then the user input function is called for the dimensions if any nets are used. If nets are used the function then calculates the total prawns allotted for the surface area per net. Both values from the values are summed and now the total number of prawns are known for the entire habitat. The user is then asked for the input of the weight and maturation period of the prawn. From this information it is then calculated when the when harvest day is, the total feed that is to be dispersed, and the new SPR that will allow the algorithm to disperse the exact amount of food. Next, the main file will ask how many times the user would like to feed the prawn (3 or 4 times) and it will ask the user what times in the day would you like to feed them. It is important the user inputs the

times in military format for the hours. An example will be given when prompted to enter the times. An example will look like “2330” which is 11:30 pm.

The section of code is where the automation and algorithm happen. Instead of explaining what each line of code is doing it is best to summarize what this section of code is executing. Once all the necessary information is gathered the user does not need to put any input and just needs to make sure that in a week or so the food container is not empty. The code takes the current prawn maturation day and gets the difference from 170 days. The difference is the amount of days it will take until the prawn are ready for harvest. It then constantly checks the time to see if it matches the user input times. The algorithm will organize the times from earliest to the latest times so it does not get stuck waiting for a time the next day and missing two or three times of feeding depending on the option you chose. After the feeding times have successfully completed, the code will then wait until the next day to start beginning the incrementing process. Prawns grow a little more than a gram a day so they weight of all the prawns need to be incremented and then the total feed can then be recalculated. From the new total feed, I could now also re-calculate a new SPR that will dispense the new total feed for the next day. This process will go on until the harvest day is reached and the program will provide an indication that the prawn are ready to be harvested and it will then terminate until the next batch is ready to start a new cycle.

4.5 Test Suite

A test suite was created for the development of the software to ensure the integrity of all the functions created for this algorithm. This is a professional technique that is used throughout industry, research, and projects. In order to test the suite, a download is required of an external Python library called Pytest using the terminal window. Initialization files need to be created inside the test directory and inside the master thesis directory because files are required to make Python treat the directories as containing packages; this is done to prevent directories with a common name, such as a string, from unintentionally hiding valid modules that occur later on the module search path [15]. To see what test passed or failed the user needs to go into the “Master Thesis Prawn Algorithm” via the terminal and run the command “pytest.” In figure 24 shows an example of when the test suite is ran to make sure the functions that were created work as intended.

```
(base) rowlandramos@Rowlands-MBP Master-Thesis-Prawn-Algorithm % pytest
===== test session starts =====
platform darwin -- Python 3.7.4, pytest-5.3.4, py-1.8.1, pluggy-0.13.1
rootdir: /Users/rowlandramos/Documents/Grad Research & Thesis/Master-Thesis-Prawn-Algorithm
collected 4 items

Project/tests/test_Project.py .... [100%]

===== 4 passed in 0.04s =====
(base) rowlandramos@Rowlands-MBP Master-Thesis-Prawn-Algorithm %
```

Figure 25. All Passed Pytest Output

In order to test the functions used in the algorithm a Python file was created called “test_Project.py” Not all functions can be tested because some functions take in user input and there is not a method to test user input. In this file test functions were created for all of the functions that can be tested. For example, in order to test the “habitatPrawn”

function there is a specific syntax that needs to take place when creating the test function. The word test with an underscore needs to be in front of the original function name itself (test_habitatPrawn). Failing to do so will result the terminal giving you messages of multiple errors. In each test function I placed explicit values that the outcome was known. Once the values were picked, the original function is called and an assert is used to ensure that the value is correct. An assert is similar to a conditional statement where if it does pass it will give you an indication that it passed just like in figure 26 or if it fails it give you a failing notification that can be seen in figure 25.

```
(base) rowlandramos@Rowlands-MBP Master-Thesis-Prawn-Algorithm % pytest
===== test session starts =====
platform darwin -- Python 3.7.4, pytest-5.3.4, py-1.8.1, pluggy-0.13.1
rootdir: /Users/rowlandramos/Documents/Grad Research & Thesis/Master-Thesis-Prawn-Algorithm
collected 4 items

Project/tests/test_Project.py F... [100%]

===== FAILURES =====
----- test_habitatPrawns -----

def test_habitatPrawns():
    """
    Function to test if the function habitatPrawns returns correct
    amount of prawns based on their dimensional habitat
    """
    length = 30
    width = 8
    height = 5

    testResult = habitatPrawns(length, width, height)
    > assert(testResult == 621)
E     assert 620 == 621

Project/tests/test_Project.py:19: AssertionError
===== 1 failed, 3 passed in 0.84s =====
```

Figure 26. Fail Example Pytest Output

In the fail example there is a lot of useful information that can be used from the test suite rather than just to see if it passes or fails. The terminal shows what test function failed which in this case was the “test_habitatPrawns” function and why it failed. You can see where the original function was called, the arguments passed in are the length, width, and height and received the value of 620 when it was supposed to get a value of 621. There

are two reasons why this could have occurred. One, the person that created the test function miscalculated and put the wrong expected value or the original function was written incorrectly in code. Either way it is best to look over both cases to make sure everything is in proper order. At the bottom of the figure it also indicates how many test functions passed and how many failed. In this example a total of one failed and three passed. Lastly, if time was a concern or if the project relied heavily on multi-threading it displays the time it took to execute the test suite which was a whopping 0.4 seconds.

5. RASPBERRY PI CONNECTION METHODS

5.1 Connecting Locally

The methods below can be used to connect to the Raspberry Pi locally to control and run the prawn algorithm. Connecting locally defines that the methods proposed have to be within the same network in order to interact with the Raspberry Pi.

Peripherals

Before explaining the different ways that the user can connect and control the Raspberry Pi, it is important to explain how to get the operating system (OS) working for the Raspberry Pi. The user could get different operating systems for the Raspberry Pi such as Ubuntu Mate, OSMC, and Windows [16]. The OS used in the feeder is the New Out of the Box Software (NOOBS) because it is known to be user friendly [17]. Once downloaded the user needs to write the NOOBS OS to a microSD card and place it in the Raspberry Pi. On initial setup the user needs to have a monitor, mouse, and keyboard to get the OS running. Once running it is highly recommended that the user updates and upgrades the OS in the terminal window so that all the software in NOOBS has the latest versions. Having peripherals is the most common way to control the Raspberry Pi but it lacks portability when working in an aquaponics environment.

Virtual Network Computing (VNC)

Another method that can be used is VNC shown in figure 27. VNC is available for almost any major OS available today [18]. VNC lets the user connect to the Raspberry Pi

from a different device such a laptop but the Raspberry Pi has to be connected to Wi-Fi prior to launching the VNC viewer on the user's personal device. VNC does not require any external peripherals to the Raspberry Pi and is intended to be used as a virtual environment. Currently VNC is supported for the following OS environments Windows, MAC, iPads, and Android tablets.

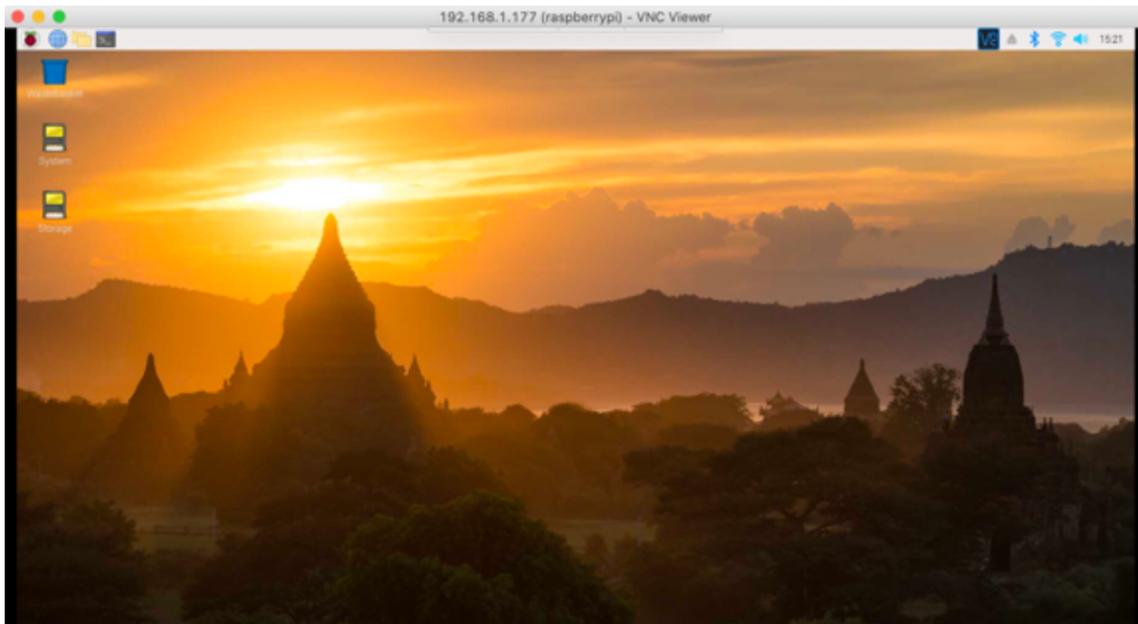


Figure 27. VNC Connection

Putty

Putty is an open source Secure Shell (SSH) client used to connect to a remote server [19]. Since the Raspberry Pi is a small single board computer is can take on an IP address and provide the user a remote server. Putty supports different network protocols for connection which includes RAW, Telnet, Rlogin, SSH, and Serial. For this thesis the type of connection used SSH to login into the Raspberry Pi via the Pi's IP address on port 22 as shown in figure 28. Once the user opens the connection a virtual terminal will prompt

the Raspberry Pi's root name and password. After that information is given the terminal of the Raspberry Pi will appear and the user can be able to navigate to the Master Prawn Algorithm to run it. Currently the best support for downloading Putty is on windows because it was specifically created for Windows OS [20]. If the user has a MAC there is way to get around this problem by third party software. First the user needs to download xCode which is an IDE for Swift created by Apple. After that the user will be asked to create a developers account where it will take the user to Apple's website. Once logged into the developer account the user needs to download a package called "Command Line Tools." Command line tools is a library for all of terminal commands that is going to be utilized in Putty. After, a xcode license is accepted by running "Sudo xcodebuild -license". Next, an open source project that is downloaded to run a version of the X.org X Window System that runs OS X [21]. Lastly, MAC PORTS is downloaded which is allows the user to use any port to connect to the conveyor system. Figure 29 demonstrates what the user will see once logged into the Raspberry Pi.

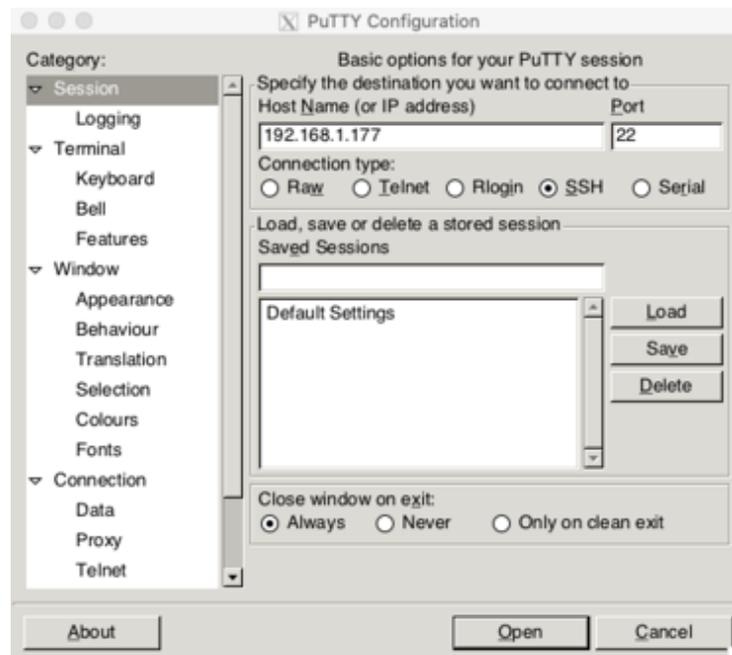


Figure 28. Putty Log In via XQuartz

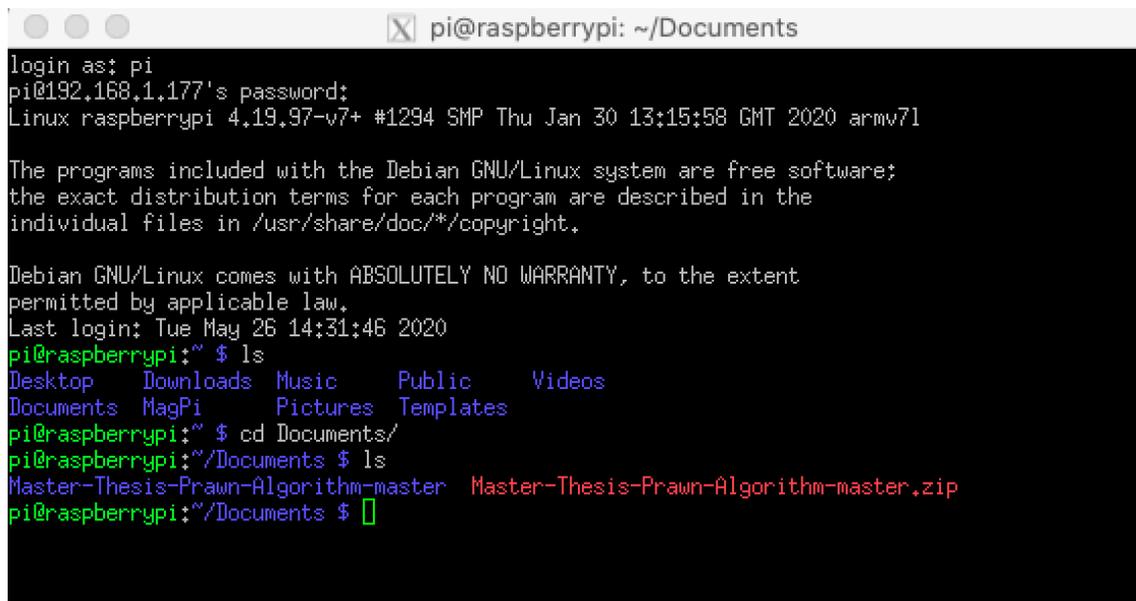


Figure 29. Putty Connection (Terminal Window)

HTML (Server Based) Research

The Raspberry Pi can be utilized as a server using Apache. Apache is an open source and free web server that is created for the public to use [22]. The Apache files are downloaded to the Raspberry Pi and an HTML file is created to create a local website. A back-end server is required to run the Python script for the conveyor system. The back-end server that is used in conjunction with Apache is Flask which is a web framework that provides tools, libraries, and technologies that allow the developer to build a web application [23]. In figure 30 it displays a basic web application that was created for the prawn algorithm. In order to create the server to communicate to the feeder system four programming languages are required which include HTML, CSS, jQuery/JavaScript, and Python.



Figure 30. Web Application Research

5.2 Connecting Remotely

The methods explained in this section detail on how to connect to the Raspberry Pi outside of the network that the Pi is in. In order to allow the connection, a setup of “Port Forwarding” is implemented. Port forwarding allows an outside host from the local network to give permission to control or manipulate a device. First, access to the local area networks (LAN) router is needed to add a port forward using the Raspberry Pi’s IP address as shown in figure 31. In order to access the router used in the LAN, the default gateway of the router is obtained and placed the default gateways’ IP address into a web browser. The browser prompts the user interface (UI) of AT&T’s router. Please refer to the specific LAN routers manual to gain access to its specific UI. The router that was accessed in this LAN was an AT&T BGW 210 router.

The image shows a web-based configuration interface for a router, specifically the 'Service Entry' section. It contains the following fields and values:

- Service Name:** A text input field containing 'Raspberry Pi'. Below it is a small example text: 'e.g. My Gaming Service'.
- Global Port Range:** Two adjacent text input fields, both containing '3389', separated by a hyphen. Below them is a small example text: 'e.g. 2000, 2000-2002'.
- Base Host Port:** A text input field containing '3389'. Below it is a small example text: 'e.g. 2000'.
- Protocol:** A dropdown menu with 'TCP/UDP' selected.
- Buttons:** A single 'Add' button is located at the bottom of the form.

Figure 31. Port Forward Setup

In figure 32 shows the setting up of a port and network protocol to use in conjunction with the Raspberry Pi. The user can choose what the service name is to be identified as but it is required to use the specific port 3389. This port is registered for

Microsoft Window Based Terminal (WBT) server and is used for Windows Remote Desktop and Remote connections using the protocol Remote Desktop Protocol (RDP) [24]. The last configuration set is the protocol which is Transmission Control Protocol/User Datagram Protocol (TCP/UDP) to allow the sending and receiving of bits from the remote device.

Hosted Applications

No Application Hosting entries have been defined

Application Hosting Entry

Service: *Raspberry Pi Service Details

Needed by Device: raspberrypi

Add

Manage Custom Services

Custom Services

Figure 32. Port Identification

In the last step of setting up the forwarding port adding specific configurations from figure 33 and assign it to the Raspberry Pi’s IP address.

Hosted Applications

Service	Ports	Device	Delete
Raspberry Pi	TCP/UDP: 3389	raspberrypi	Delete

Figure 33. Port Forward Finished

For this router UI, this is what it looks like when the port forward is properly completed. To control the Raspberry Pi remotely the public IP address of the LAN’s router was obtained. For security reasons the public IP address will not be shared, but any user

can find it by typing “what is my IP?” into Google. It should not start off with 192.168.X.X or 10.0.X.X because those are private IP addresses. An example of what it could start off as is 71.X.X.X.

Remote Desktop Mobile (App)

Once all of Port Forwarding is completed there is a free application that is available to download that which is called Remote Desktop Mobile for Apple, Microsoft, and Android devices. Once downloaded, the forwarding port and Raspberry Pi’s IP address within the application is setup within the application. To setup the forwarding port just hit the “+” button in the app and place the public IP address with the RDP port in this syntax “71.X.X.X:3389” and hit save [25]. The same procedure is done for the Raspberry Pi’s IP address and hit save. To login remotely the user clicks on the Raspberry Pi’s saved configuration in the app which it will then prompt a login screen. If everything was setup correctly it should look like figure 34.

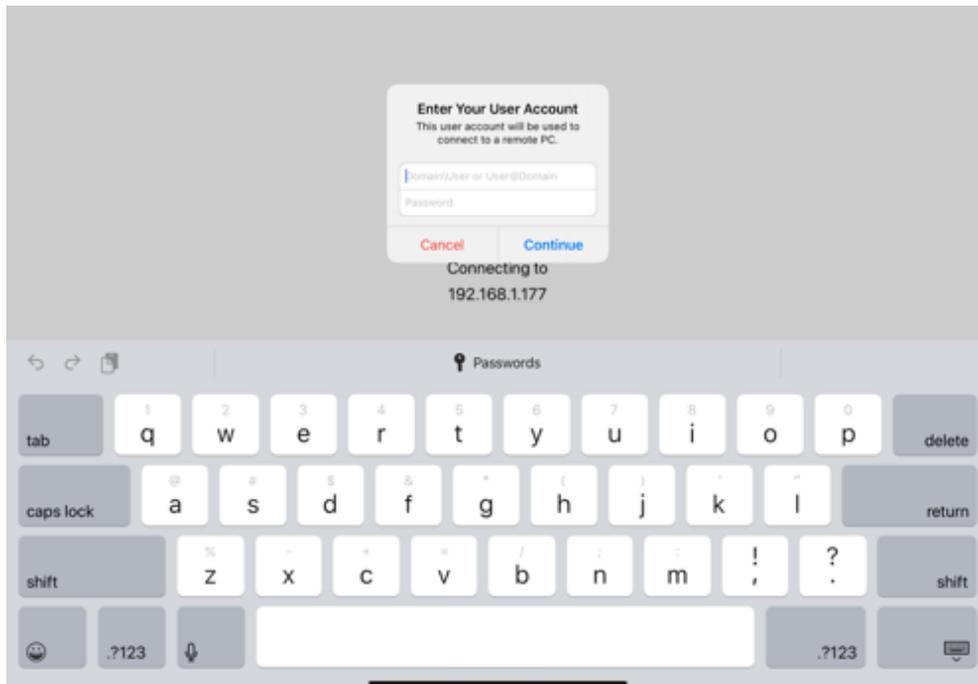


Figure 34. Log-In Remote Desktop App (iPad)

After logging into the Raspberry Pi, the user can control the NOOBS interface with the touch of the fingers. By being able to control the conveyor system remotely allows anyone to check the status of the system and make any adjustments anywhere and anytime. Another huge advantage to controlling a device like this is the portability. A laptop or any traditional computing externals is not required to control the conveyor system. Any user can use an iPhone or any Android phone to control the system as well. In figure 35 shows the Raspberry Pi OS and shows a Safari window opened from the iPad to show that it is connected through an iPad.

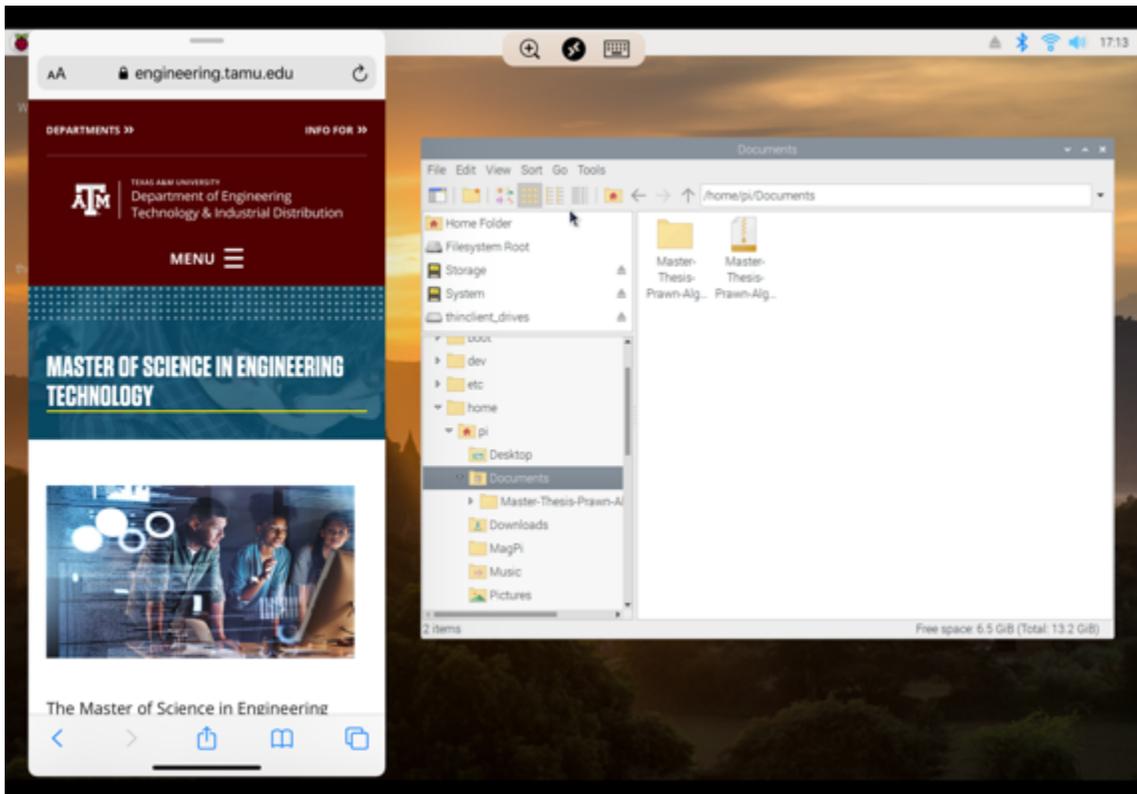


Figure 35. Remotely Connected via iPad

HTML (Server Based) Research

The conveyor system can be controlled with the front and back end server created remotely as well as locally. This approach cannot run the algorithm because the website is not fully developed. Once the website is fully developed the user can just open a web browser from any device and be able to reach the Raspberry Pi's IP address and see the status of the algorithm in real time.

6. RESULTS

6.1 Experiment Setup

For the experimental setup, prawn food was not used in the results due to COVID-19 restricting resources. A dog food called American Journey Salmon and Sweet Potato 12lb bag was used. This experiment is going to demonstrate the differences and similarities that each microstepping index that the stepper motor is capable of. At full step the motor produces the most amount of torque but provides least step angle which is a 1.8° . The highest micro step the stepper motor is capable of is $32\mu S$ which torque is significantly lost but the stepper motor can be controlled $1/32$ of 1.8° . The calibration ratio that was used for this experiment is 200 steps per revolution for every 35 grams of food dispensed. Five different sample food amounts are going to be tested for each microstepping index which are 50 grams, 100 grams, 150 grams, 200 grams, and 250 grams.

6.2 Statistics

Table 6 shows how much food was weighed at each microstepping index compared to the ideal weight. For each revolution the index was randomly picked so the results would not only focus on one specific index at a time throughout the experiment. This process ensures that the variability of the algorithm is tested when given random microstepping indexes. The results did not seem to have a trend because the how the food size took on a different pattern each time making the results inconsistent. We predict that the percent errors and standard deviation will be lower with prawn food because it takes

the shape similar to rice, which will allow the conveyor system to dispense the food more smoothly.

Table 6. Food Dispensed (Actual VS Ideal)

	Revs	Full	1/2	1/4	8uS	16uS	32uS	Ideal
Food Dispensed (g)	1	59	47	47	51	52	54	50
Food Dispensed (g)	2	91	97	107	95	93	96	100
Food Dispensed (g)	3	142	134	133	148	145	158	150
Food Dispensed (g)	4	203	189	199	185	204	195	200
Food Dispensed (g)	5	249	250	259	252	241	249	250

Table 7. Percentage Errors and Standard Deviation

	Revs	Full	1/2	1/4	8uS	16uS	32uS
Percentage Error (%)	1	18	6	6	2	4	8
Percentage Error (%)	2	9	3	7	5	7	4
Percentage Error (%)	3	5.3333	10.67	11.3	1.3	3.33	5.33
Percentage Error (%)	4	1.5	5.5	0.5	7.5	2	2.5
Percentage Error (%)	5	0.4	0	3.6	0.8	3.6	0.4
Standard Deviation (σ)		7.097	3.948	4.03	2.8	1.84	2.87

Table 7 shows the percent errors (PE) for each index of the stepper motor for every revolution. The largest PE was at the quarter index and the least PE was at the half index with dispensing the exact amount of food which was 250 grams. The highest standard deviation was at full step at 7.097 and the smallest standard deviation was 1.84 at the index of 16 μ S.

7. LIMITATIONS AND STRENGTHS

7.1 Limitations and Strengths

There are limitations that this algorithm can do such as changing the values of the algorithm in real-time. A real-time example is when a user from a device opens up the application Google maps; their location is represented by a blue dot and their location mimics where the user is at in the world. This happens because data from their device is constantly sending and receiving information to satellites, making their location update in real-time. What makes this feature possible is having front and back end servers to be able to read and write data to one another. The current algorithm cannot update values such as feed percentages, habitat dimensions, or the steps per revolution (SPR) value in real-time. The user must exit the program and put all of the new information again and update the maturation period.

Another limitation that this system currently has is the use of exposed wires connecting the Raspberry Pi, Stepper Motor Driver, and the Stepper Motor. Humidity levels can get up to one-hundred percent in an aquaponic environment and having exposed wires can cause internal corrosion, short circuit the system, or cause the electrical devices to catch fire. Another limitation of having exposed wire connections is that it creates a more difficult setup when trying to mount all of the electrical components onto a surface or into a water proof container. Wire connections can be disconnected by accident or by nature making the automatic prawn feeder prone to malfunctions during the feeding process. Lastly, wires can be expensive depending on the length needed to properly setup

the prawn feeder and having longer wires throughout the aquaponic environment creates more vulnerability of the wires getting damaged.

Strengths that this algorithm provides is automatically dispensing the correct amount of food every time no matter the conveyor system dimensions. It alleviates weighing out the food three to four times a day just to ensure the prawn are getting the right amount of food. It also prevents farmers from overfeeding or underfeeding prawn and killing a percentage of the prawn density within their habitat. The algorithm dismisses any conveyor system dimensions because it has a calibration file that is based off of the body mass of an individual prawn and how much food it dispenses from the revolutions of the stepper motor.

8. CONCLUSION AND FUTURE WORK

8.1 Conclusion

Developing this algorithm creates a technology for the aquaponics community and can change the current feeding techniques used to feed prawn from a prawn based aquaponic system. Not only can farmers benefit from this algorithm but research from different agriculture organizations can be conducted using different control variables such as food percentages at which to feed the prawn, time effects on prawn, and the amount of times per day the prawn eat. No matter what the control variable is the algorithm can be manipulated in such a way that it will fulfill the research purpose. For instance, if researchers or industry want to have the control variable to be the percentage of the body mass of the prawn, they can have three separate habitats all with different percentages using the same algorithm. Another purpose for this algorithm is for those who want to use it for humanitarian purposes in third world countries. This algorithm enables people that have no experience in software development and farming to download the algorithm file and run it with their own conveyor system. This technology would be able to provide an accessible prawn farming practice throughout the world.

8.2 Future Work

Though this algorithm is a great start in the right direction for the aquaponics community there is future work that can be done. Creating an application, such as a website, would create a more user-friendly experience with animations when inputting values for the algorithm. A developer can create a user interface on the website with input text-field boxes and submission buttons detailing what specific input the user wants to use. Unlike running a Python file, most people have interacted with a web browser before, so navigating through a website will feel more natural and prawn farmers can easily become familiar with the graphical user interface (GUI).

Another contribution that can be done in the future is creating a printed circuit board (PCB) for the prawn feeder's electrical and electromechanical components. A major plus to having a PCB is it will allow everything to be plug and play to get the system going. No wires will be required and there will be no need having to figure out what connection goes where. Having the PCB will also allow everything to be easily secured in a water proof container because the PCB will have four holes in each corner so that the user can put screws through and mount it into the container. This would be the most reliable way to allow the electrical components to communicate with one another.

Lastly, adding a motion sensor combined with a Raspberry Pi Camera Module will create a feedback loop for the conveyor system. This feedback loop is for food malfunctions that the prawn feeder might encounter. The motion sensor will allow the prawn farmer get an alert from an email entailing that the prawn feeder is sensing a lower count than usual or that it is not sensing anything at all. From this alert, the camera module

will allow the prawn farmer to identify what is causing the issue by using the prawn feeder's internet of things (IoT) capabilities to connect locally or remotely.

REFERENCES

- [1] Ritchie, Hannah and Roser, Max (2019). "Age Structure". *Published online at OurWorldInData.org*. Retrieved from: '<https://ourworldindata.org/age-structure>' Accessed 1 May, 2020.
- [2] Olayanju, Julia (2019). "Plant-Based Meat Alternatives: Perspectives on Consumer Demands and Future Directions". Retrieved from: '<https://www.forbes.com/sites/juliabolayanju/2019/07/30/plant-based-meat-alternatives-perspectives-on-consumer-demands-and-future-directions/#6cf4849b6daa>' Accessed 1 May, 2020.
- [3] Bashi, Zafer, McCullough, Ryan, Ong, Liane, Ramirez, Miguel (2019). "Alternative Proteins: The race for market share is on". Retrieved from: '[https://www.mckinsey.com/industries/agriculture/our-insights/alternative-proteins-the-race-for-market-share-is-on#:~:text=Alternatives%20are%20protein%2Drich%20ingredients,based%20sources%20\(Exhibit%203\).](https://www.mckinsey.com/industries/agriculture/our-insights/alternative-proteins-the-race-for-market-share-is-on#:~:text=Alternatives%20are%20protein%2Drich%20ingredients,based%20sources%20(Exhibit%203).)' Accessed 1 May, 2020.
- [4] Food and Agriculture Organization of the United Nations (2014). "A Common Food". Retrieved from: '<http://www.fao.org/edible-insects/84742/en/>' Accessed 1 May, 2020.

[5] Kolb, Elzy (2019). “Countries with the top 20 fastest-growing populations”. Retrieved from: ‘<https://www.usatoday.com/story/money/2019/07/10/world-population-day-fastest-growing-countries-guinea-chad-mali/39584997/>’ Accessed 1 May, 2020.

[6] Martins, Fabricio, Ballester, Eduardo Luis Cupertino (2017). “Determining safe levels of ammonia and nitrite for shrimp culture”. Retrieved from: ‘<https://www.aquaculturealliance.org/advocate/determining-safe-levels-of-ammonia-and-nitrite-for-shrimp-culture/>’ Accessed 1 May, 2020.

[7] Mitra, Gopa, Mukhopadhyay, P.K. (2005). “Nutrition and Feeding in Freshwater Prawn (*Macrobrachium rosenbergii*) Farming”. Retrieved from: ‘<https://agrilibecdn.tamu.edu/fisheries2/files/2013/09/Nutrition-and-Feeding-in-Freshwater-Prawn-Macrobrachium-rosenbergii-Farming.pdf>’ Accessed 1 May, 2020.

[8] New, Michael (1980). “The Diet of Prawns”. Retrieved from: ‘<http://www.fao.org/3/AB915E/AB915E00.htm#:~:text=In%20nature%2C%20prawns%20eat%20frequently,of%20the%20dietary%20stability%20problems.>’ Accessed 1 May, 2020.

[9] Seafood Source (2014). “Shrimp, Freshwater”. Retrieved from: ‘<https://www.seafoodsource.com/seafood-handbook/shellfish/shrimp-freshwater>’ Accessed 1 May, 2020.

[10] LIVE Aquaponics (2020). “Handling Freshwater Shrimp/Prawns”. Retrieved from: ‘<https://liveaquaponics.com/blogs/news/how-to-grow-h-u-g-e-freshwater-prawns>’ Accessed 1 May, 2020.

[11] DISPLAYS2GO (2019). “The Many Forms of Plastic”. Retrieved from: ‘<https://www.displays2go.com/Article/The-Many-Forms-Clear-Plastic-5>’ Accessed 1 May, 2020.

[12] Texas Instruments (2014). “DRV8825 Stepper Motor Controller IC”. Retrieved from: ‘<https://www.ti.com/lit/ds/symlink/drv8825.pdf>’ Accessed 1 May, 2020.

[13] Changzhou Songyang Machinery & Electronics New Technic Institute (2013). “High Torque Hybrid Stepping Specifications”. Retrieved from: ‘<https://www.pololu.com/file/0J672/SY57STH56-1006A.pdf>’ Accessed 1 May, 2020.

[14] Raspberry Pi Forum (2013). “Difference between GPIO.setmode(BCM) and GPIO.setmode(BOARD)”. Retrieved from: ‘<https://www.raspberrypi.org/forums/viewtopic.php?t=34273>’ Accessed 1 May, 2020.

[15] Python Tips (2013). “What is `__init__.py`?”. Retrieved from: ‘https://pythontips.com/2013/07/28/what-is-__init__.py/’ Accessed 1 May, 2020.

[16] Raspberry Pi (2020). “Downloads”. Retrieved from:

‘<https://www.raspberrypi.org/downloads/>’ Accessed 1 May, 2020.

[17] Raspberry Pi (2020). “NOOBS”. Retrieved from:

‘<https://www.raspberrypi.org/downloads/noobs/>’ Accessed 1 May, 2020.

[18] REALVNC (2020). “VNC CONNECT”. Retrieved from:

‘<https://www.realvnc.com/en/connect/download/viewer/macos/>’ Accessed 1 May, 2020.

[19] Shiva (2017). “PuTTY – 30 Useful Putty Commands for Beginners”. Retrieved

from: ‘<https://www.fastwebhost.in/blog/putty-30-useful-putty-commands-for-beginners/>’

Accessed 1 May, 2020.

[20] PuTTY (2020). “Download PuTTY”. Retrieved from: ‘<https://www.putty.org/>’

Accessed 1 May, 2020.

[21] XQuartz (2020). “Quick Download”. Retrieved from: ‘<https://www.xquartz.org/>’

Accessed 1 May, 2020.

[22] G., Domantas (2020). “What is Apache? An In-Depth Overview of Apache Web Server”. Retrieved from: ‘<https://www.hostinger.com/tutorials/what-is-apache>’ Accessed 1 May, 2020.

[23] Das, Kushal (2020). “Introduction to Flask”. Retrieved from: ‘<https://pymbook.readthedocs.io/en/latest/flask.html>’ Accessed 1 May, 2020.

[24] SpeedGuide (2020). “Port 3389 Details”. Retrieved from: ‘<https://www.speedguide.net/port.php?port=3389>’ Accessed 1 May, 2020.

[25] Circuit Basics (2015). “How to access the Raspberry Pi Desktop with a Remote Connection”. Retrieved from: ‘<https://www.circuitbasics.com/access-raspberry-pi-desktop-remote-connection/>’ Accessed 1 May, 2020.

APPENDIX A

FILENAMES WITH CODE

A.1 MainFile.py

```
# Master Thesis: Prawn Algorithm
# Author: Rowland Andrew Ramos

import math
import time
from feederFunctions import *
from calibrationFunction import *
from datetime import date, datetime, timedelta

global spr, foodDispensed, habitatTotal, netTotal, weight, startingDay
global timeArray, fileName, numberOfTimes

def mainMenu():
    # Displaying Menu when file is ran
    print("      ***** IMPORTANT: PLEASE READ ***** ")
    print(" YOU MUST INPUT ALL INFORMATION BEFORE STARTING
ALGORITHM!\n\n")
    print("1. Input calibration ratio")
    print("2. Input habitat dimensions")
    print("3. Input net dimensions")
    print("4. Input prawn weight and maturation period")
    print("5. Input number of times per day you want to feed prawn")
    print("6. File name")
    print("7. Start algorithm")
    print("8. Exit")

try:
    # Getting user input for menu option
    menuInput = int(input("Enter number for choice.\n->"))
    # Option 1 - Calibration Input
    if menuInput == 1:
        # Food input must be in grams
        global spr
        global foodDispensed
        spr , foodDispensed = calRatio()
        mainMenu()
```

```

# Option 2 - Habitat Dimensions Input
elif menuInput == 2:
    # Getting user input for dimensions of habitat
    habArray = habInputs()
    global habitatTotal
    habitatTotal = habitatPrawns(habArray[0], habArray[1], habArray[2])
    mainMenu()

# Option 3 - Net Dimension Input
elif menuInput == 3:
    # Getting user input for dimensions of nets
    netArray = netInputs()
    global netTotal
    netTotal = netPrawns(netArray[0], netArray[1], netArray[2])
    mainMenu()

# Option 4 - Prawn wieght and maturation period input
elif menuInput == 4:
    # Getting prawn weight and starting day for feeding
    print("      ***** IMPORTANT: PLEASE READ *****      ")
    print("Most prawn production starts at 30 or 45 days and usually weighs around
0.15 grams per prawn.")
    userInput = input("If weight is vastly different please enter -> other, if not please
enter -> normal.\n->")
    userInput = userInput.lower()
    while userInput != "normal" and userInput != "other":
        print("Invalid input!")
        userInput = input("If weight is vastly different please enter -> other, if not
please enter -> normal.\n->")
    userInput = userInput.lower()
    global weight
    global startingDay
    weight, startingDay = prawnWeight(userInput)
    mainMenu()

# Option 5 - Number of time to feed in a day
elif menuInput == 5:
    global numberOfTimes
    numberOfTimes = input("How many times would you like to feed the prawn per
day? 3 or 4?\n->")
    while numberOfTimes != "3" and numberOfTimes != "4":
        numberOfTimes = input("Invalid input! Must be 3 or 4 times?\n->")
    global timeArray
    timeArray = timeInputs(numberOfTimes)

```

```

mainMenu()

# Option 6 - Filname Input
elif menuInput == 6:
    global fileName
    fileName = input("What would you like the file named to be called?\n->")
    mainMenu()

# Option 7 - Algorithm Start
elif menuInput == 7:
    # Calculating total prawns
    totalPrawns = netTotal + habitatTotal
    # Getting total feed in grams for all prawns at 8% of their mass
    feedTotal = totalFeed(totalPrawns, weight)
    # Rounding up on the feed total
    roundedfeedTotal = math.ceil(feedTotal)
    print("The total feed that will be dispensed is: ", roundedfeedTotal, "grams")
    # Gathering date information and SPR
    harvestDelta = 170 - startingDay
    currentDate = date.today()
    delta = timedelta(days=harvestDelta)
    harvestDate = currentDate + delta
    newRevolution = SPR(spr, foodDispensed, roundedfeedTotal)
    if numberOfTimes == "3":
        prawnFile = open(fileName + ".txt", "w")
        title = "Feed Total (g) Current Date Current Time\n"
        prawnFile.write(title)
        while currentDate != harvestDate:
            currentTime = datetime.now().strftime("%H%M")
            while currentTime != timeArray[0]:
                # Do nothing
                currentTime = datetime.now().strftime("%H%M")
            motor(newRevolution)
            print("Food amount dispensed in grams:", roundedfeedTotal, "Date: ",
currentDate , "Time:", currentTime)
            prawnFile.write(str(roundedfeedTotal)+"          " +str(currentDate)+"
"+str(currentTime)+"\n")
            prawnFile.close()
            prawnFile = open(fileName+".txt", "a")
            while currentTime != timeArray[1]:
                # Do nothing
                currentTime = datetime.now().strftime("%H%M")
            motor(newRevolution)

```

```

        print("Food amount dispensed in grams:", roundedfeedTotal, "Date: ",
currentDate , "Time:", currentTime)
        prawnFile.write(str(roundedfeedTotal)+"          " +str(currentDate)+"
"+str(currentTime)+"\n")
        prawnFile.close()
        prawnFile = open(fileName+".txt", "a")
        while currentTime != timeArray[2]:
            # Do nothing
            currentTime = datetime.now().strftime("%H%M")
            motor(newRevolution)
            print("Food amount dispensed in grams:", roundedfeedTotal, "Date: ",
currentDate , "Time:", currentTime)
            prawnFile.write(str(roundedfeedTotal)+"          " +str(currentDate)+"
"+str(currentTime)+"\n")
            prawnFile.close()
            prawnFile = open(fileName+".txt", "a")
            currentTime = datetime.now().strftime("%H %M %S")
            while currentTime != "23 59 59":
                # Waiting for midnight to get new day
                currentTime = datetime.now().strftime("%H %M %S")
            time.sleep(5)
            #Getting new date
            currentDate = date.today()
            # Incrementing Prawn weight
            weight = weight + 1.214
            feedTotal = totalFeed(totalPrawns, weight)
            roundedfeedTotal = math.ceil(feedTotal)
            newRevolution = SPR(spr, foodDispensed, roundedfeedTotal)
elif numberOfTimes == "4":
    prawnFile = open(fileName + ".txt", "w")
    title = "Feed Total (g) Current Date Current Time\n"
    prawnFile.write(title)
    while currentDate != harvestDate:
        currentTime = datetime.now().strftime("%H %M %S")
        while currentTime != timeArray[0]:
            # Do nothing
            currentTime = datetime.now().strftime("%H%M")
            motor(newRevolution)
            print("Food amount dispensed in grams:", roundedfeedTotal, "Date: ",
currentDate , "Time:", currentTime)
            prawnFile.write(str(roundedfeedTotal)+"          " +str(currentDate)+"
"+str(currentTime)+"\n")
            prawnFile.close()
            prawnFile = open(fileName+".txt", "a")

```

```

while currentTime != timeArray[1]:
    # Do nothing
    currentTime = datetime.now().strftime("%H%M")
    motor(newRevolution)
    print("Food amount dispensed in grams:", roundedfeedTotal, "Date: ",
currentDate , "Time:", currentTime)
    prawnFile.write(str(roundedfeedTotal)+"      " +str(currentDate)+"
"+str(currentTime)+"\n")
    prawnFile.close()
    prawnFile = open(fileName+".txt", "a")
    while currentTime != timeArray[2]:
        # Do nothing
        currentTime = datetime.now().strftime("%H%M")
        motor(newRevolution)
        print("Food amount dispensed in grams:", roundedfeedTotal, "Date: ",
currentDate , "Time:", currentTime)
        prawnFile.write(str(roundedfeedTotal)+"      " +str(currentDate)+"
"+str(currentTime)+"\n")
        prawnFile.close()
        prawnFile = open(fileName+".txt", "a")
        while currentTime != timeArray[3]:
            # Do nothing
            currentTime = datetime.now().strftime("%H%M")
            motor(newRevolution)
            print("Food amount dispensed in grams:", roundedfeedTotal, "Date: ",
currentDate , "Time:", currentTime)
            prawnFile.write(str(roundedfeedTotal)+"      " +str(currentDate)+"
"+str(currentTime)+"\n")
            prawnFile.close()
            prawnFile = open(fileName+".txt", "a")
            currentTime = datetime.now().strftime("%H %M %S")
            while currentTime != "23 59 59":
                # Waiting for midnight to get new day
                currentTime = datetime.now().strftime("%H %M %S")
            time.sleep(5)
            #Getting new date
            currentDate = date.today()
            # Incrementing Prawn weight
            weight = weight + 1.214
            feedTotal = totalFeed(totalPrawns, weight)
            roundedfeedTotal = math.ceil(feedTotal)
            newRevolution = SPR(spr, foodDispensed, roundedfeedTotal)
else:
    print("Invalid input!")

```

```

    # Print statement indicating program is finished and prawn are ready for harvest
    print("Prawns are ready for harvest!")
    prawnFile.close()
    exit
# Option 8 - Exit Program
elif menuInput == 8:
    print("PROGRAM TERMINATED!")
    exit
# Conditional to prevent wrong input
else:
    print("Invalid input! Must be an input 1-8!\n\n")
    mainMenu()
except ValueError:
    print("Invalid input! Must be an input 1-8!\n\n")
    mainMenu()
# Menu function called
mainMenu()

```

A.2 feederFunctions.py

```

# Master Thesis: Prawn Algorithm
# Author: Rowland Andrew Ramos

import math
from datetime import date, datetime, timedelta

def habInputs():
    """Function to get user input for habitat dimensions"""
    print("Please enter dimensions of your pool in feet.")
    print("No decimals. Round up or down for dimension values.\n")

    dimArray = []
    numArray = ['0','1','2','3','4','5','6','7','8','9',
                '10','11','12','13','14','15','16','17',
                '18','19','20','21','22','23','24','25',
                '26','27','28','29','30','31','32','33',
                '34','35','36','37','38','39','40','41',
                '42','43','44','45','46','47','48','49',
                '50']

    while len(dimArray) != 1:
        lengthNum = input("Enter length of habitat:\n->")

```

```

    if lengthNum not in numArray:
        print("Invalid! Must be an whole number value.\n")

    elif lengthNum in numArray:
        lengthNum = int(lengthNum)
        dimArray.append(lengthNum)

while len(dimArray) != 2:
    widthNum = input("Enter width of habitat:\n->")
    if widthNum not in numArray:
        print("Invalid! Must be an whole number value.\n")

    elif widthNum in numArray:
        widthNum = int(widthNum)
        dimArray.append(widthNum)

while len(dimArray) != 3:
    heightNum = input("Enter height of habitat:\n->")
    if heightNum not in numArray:
        print("Invalid! Must be an whole number value.\n")

    elif heightNum in numArray:
        heightNum = int(heightNum)
        dimArray.append(heightNum)

return dimArray

def habitatPrawns(length, width, height):
    """
    Function to calculate the prawn allowed in the habitat
    dimension length, width, height are in feet
    """
    starportWalls = (length * height) * 2
    deckWall = length * width
    foreaftWalls = (width * height) * 2
    habPrawns = starportWalls + deckWall + foreaftWalls
    return habPrawns

def netInputs():
    """Function to get user input for net dimensions"""
    print("Please enter dimension of your net(s) in feet.\n")
    print("No decimals. Round up or down for dimension values.\n")

```

```

netArray = []
numArray = ['0','1','2','3','4','5','6','7','8','9',
            '10','11','12','13','14','15','16','17',
            '18','19','20','21','22','23','24','25',
            '26','27','28','29','30','31','32','33',
            '34','35','36','37','38','39','40','41',
            '42','43','44','45','46','47','48','49',
            '50']

while len(netArray) != 1:
    lengthNum = input("Enter length of net:\n->")
    if lengthNum not in numArray:
        print("Invalid! Must be an whole number value.\n")

    elif lengthNum in numArray:
        lengthNum = int(lengthNum)
        netArray.append(lengthNum)

while len(netArray) != 2:
    widthNum = input("Enter width of net: ")
    if widthNum not in numArray:
        print("Invalid! Must be an whole number value.\n")

    elif widthNum in numArray:
        widthNum = int(widthNum)
        netArray.append(widthNum)

while len(netArray) != 3:

    numofNets = input("Enter number of net(s):\n->")
    if numofNets not in numArray:
        print("Invalid! Must be an whole number value.\n")

    elif numofNets in numArray:
        numofNets = int(numofNets)
        netArray.append(numofNets)
return netArray

```

```

def netPrawns(length, height, numofNets):
    """
    Function calculates additional allotted prawns by using 'n'

```

```

number of nets
"""
netTotal = length * height * numofNets
return netTotal

def prawnWeight(option):
    """
    Function to determine prawn wieght and starting day of prawn life. Weight is
    grams
    """
    if option == "normal":
        prawnWeight = 0.15;
        startingDay = 30;
        return prawnWeight, startingDay
    elif option == "other":
        prawnWeight = float(input("Enter the weight of an individual prawn in grams.\n-
>"))
        startingDay = int(input("Enter the number of days old the prawn are.\n->"))
        return prawnWeight, startingDay

# Functions below are used when the cycle begins until the prawns are ready for harvest

def totalFeed(totalPrawns, weight):
    """
    Function to calculate how much food should be dispensed in grams at 8%
    """
    # The max a prawn can weigh is 600 Grams = 1.3 lbs = 21.2 Ounces
    # Typically a prawn is 6-7 inches 72.57 Grams = 0.16 lbs = 2.56 Ounces
    # Another study showed it can weigh 113.4-170.1 Grams = 4-6 Ounces = 0.25-
    0.375lbs (Using this)
    feedTotal = (totalPrawns * weight) * 0.08
    return feedTotal

def SPR(spr, foodDispensed, feedTotal):
    """
    Function to determine new steps per revolution for new feed total
    """
    newRevolution = spr * feedTotal
    newRevolution = newRevolution / foodDispensed
    roundRevolution = math.ceil(newRevolution)
    return roundRevolution

```

```

def calRatio():
    """Function to get user input for SPR to average food dispensed during calibration"""
    print("It is highly recommended that you run the calibration at minimum of ten times
before entering this information!")
    spr = int(input("Enter the SPR value you used in callibration\n->"))
    foodDispensed = int(input("Enter average food value from SPR in grams.\n->"))

    return spr, foodDispensed

```

```

def timeInputs(numberTimes):
    """Function to get user input for feed times in day"""
    timeArray = []
    print("Type time in military format. Example: 6:30PM would be 1830.\n")
    print("IMPORTANT: Start from the morning to evening")

```

```

if numberTimes == "3":

```

```

    firstTime = input("Enter the first time in correct format.\n->")
    timeArray.append(firstTime)

```

```

    while len(timeArray[0]) != 4:

```

```

        print("Must be four numbers.")

```

```

        timeArray[0] = input("Enter the first time in correct format.\n->")

```

```

    secondTime = input("Enter the second time in correct format.\n->")

```

```

    timeArray.append(secondTime)

```

```

    while len(timeArray[1]) != 4:

```

```

        print("Must be four numbers.")

```

```

        timeArray[1] = input("Enter the second time in correct format.\n->")

```

```

    thirdTime = input("Enter the third time in correct format.\n->")

```

```

    timeArray.append(thirdTime)

```

```

    while len(timeArray[2]) != 4:

```

```

        print("Must be four numbers.")

```

```

        timeArray[2] = input("Enter the third time in correct format.\n->")

```

```

    timeArray.sort()

```

```

    return timeArray

```

```

elif numberTimes == "4":

```

```

firstTime = input("Enter the first time in correct format.\n->")
timeArray.append(firstTime)
while len(timeArray[0]) != 4:
    print("Must be four numbers.")
    timeArray[0] = input("Enter the first time in correct format.\n->")

secondTime = input("Enter the second time in correct format.\n->")
timeArray.append(secondTime)
while len(timeArray[1]) != 4:
    print("Must be four numbers.")
    timeArray[1] = input("Enter the second time in correct format.\n->")

thirdTime = input("Enter the third time in correct format.\n->")
timeArray.append(thirdTime)
while len(timeArray[2]) != 4:
    print("Must be four numbers.")
    timeArray[2] = input("Enter the third time in correct format.\n->")

forthTime = input("Enter the fourth time in correct format.\n->")
timeArray.append(forthTime)
while len(timeArray[3]) != 4:
    print("Must be four numbers.")
    timeArray[3] = input("Enter the fourth time in correct format.\n->")

timeArray.sort()
return timeArray

```

A.3 calibrationFile.py

```

# Master Thesis: Prawn Algorithm
# Author: Rowland Andrew Ramos

import math
from time import sleep
import RPi.GPIO as GPIO
from time import perf_counter

def motor(spr):

```

```

"""
Function used to calibrate speed of motor in order
to dispense correct amount of food
"""
GPIO.setwarnings(False)

# GPIO pins for direction and step
direction = 20
step = 21

# Orientation of motor
cw = 1
ccw = 0

# Setting GPIO pins as outputs
GPIO.setmode(GPIO.BCM)
GPIO.setup(direction, GPIO.OUT)
GPIO.setup(step, GPIO.OUT)
GPIO.output(direction, ccw)

# Setup pins so I can use microstepping
mode = (14, 15, 18)
GPIO.setup(mode, GPIO.OUT)

# Initializing microstepping
stepMode = {'Full': (0,0,0),
            '1/2': (1,0,0),
            '1/4':(0,1,0),
            '8us': (1,1,0),
            '16us': (0,0,1),
            '32us': (1,0,1)}

GPIO.output(mode, stepMode['1/2'])

# spr -> Steps per revolution
# 100 spr = 17-18 grams of food
stepsperRev = spr
# This is one revolution
stepCount = stepsperRev * 2

# Delay is (1/200) divided by revolution
delay = 0.005 / 2

initalTime = perf_counter()

```

```

for x in range(stepCount):
    GPIO.output(step, GPIO.HIGH)
    sleep(delay)
    GPIO.output(step, GPIO.LOW)
    sleep(delay)

finalTime = perf_counter()
elapsedTime = finalTime - initialTime
return elapsedTime

```

A.4 test_Project.py

```

# Master Thesis: Prawn Algorithm
# Author: Rowland Andrew Ramos

import pytest
from pytest import approx
from Project.feederFunctions import *

def test_habitatPrawns():
    """
    Function to test if the function habitatPrawns returns correct
    amount of prawns based on their dimensional habitat
    """
    length = 30
    width = 8
    height = 5

    testResult = habitatPrawns(length, width, height)
    assert(testResult == 621)

def test_netPrawns():
    """
    Function to test if the function netPrawns returns correct
    amount of prawns based on the net dimensions
    """
    length = 28
    height = 5
    numofNets = 4

```

```

testResult = netPrawns(length, height, numofNets)
assert(testResult == 560)

def test_totalFeed():
    """
    Function to test if the function totalFeed returns correct amount of feed
    in grams
    """
    totalPrawns = 1520
    weight = 0.15

    testResult = totalFeed(totalPrawns, weight)
    assert(testResult == approx(18.24))

def test_SPR():
    """
    Function to test if the function SPR (Steps per revolution) returns new SPR
    based on the amount of total feed
    """
    spr = 100
    foodDispensed = 17
    feedTotal = 112

    testResult = SPR(spr, foodDispensed, feedTotal)
    assert(testResult == 659)

```

A.5 prawnApp.conf

```

<VirtualHost *:80>
    ServerName prawnpi
    WSGIDaemonProcess prawnapp user=pi group=www-data threads=5
    WSGIScriptAlias /prawnapp /var/www/prawnapp/prawnapp.wsgi
    <Directory /var/www/prawnapp>
        WSGIProcessGroup prawnapp
        WSGIApplicationGroup &{GLOBAL}
        Require all granted
    </Directory>
</VirtualHost>

```

A.6 prawnApp.wsgi

```

#!/usr/bin/env python3

```

```

activate_this = '/var/www/prawnapp/venv/bin/activate_this.py'
with open(activate_this) as file_:
    exec(file_.read(), dict(__file__=activate_this))

import sys
sys.path.insert(0, '/var/www/prawnapp')

from prawnapp import app as application

```

A.7 mainProject.py (Server Base)

```

# Master Thesis: Algorithm for Prawns
# Author: Rowland Andrew Ramos
from flask import Flask, request, jsonify
from time import sleep
import RPi.GPIO as GPIO

from feederFunctions import *

GPIO.setwarnings(False)

app = Flask(__name__)

# GPIO pin configurations
direction = 20
step = 21
cw = 1
ccw = 0

# Setting GPIO pins as outputs
GPIO.setmode(GPIO.BCM)
GPIO.setup(direction, GPIO.OUT)
GPIO.setup(step, GPIO.OUT)
GPIO.output(direction, ccw)

# Setup pins so I can use microstepping
mode = (14, 15, 18)
GPIO.setup(mode, GPIO.OUT)

# {{url}}/motor?status=on
@app.route('/', methods=['GET', 'POST'])

```

```

def motor():
    status = request.args.get('status')
    if status == "on":
        # calibration(mode, step)
        calibration(mode, step, 2000)
        return jsonify({"message": "Motor turned on!"})
    elif status == "off":
        calibration(mode, step, 0)
        return jsonify({"message": "Its working!"})
    else:
        return jsonify({"message": "Invalid response!"})

```

A.8 index.html

```

<!DOCTYPE hmtl>
<html>

<center><font size = '+5' color='maroon'>Master Thesis: Prawn Algorithm </font$

    <head>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.$
        <script src="prawnScript.js"></script>
        <link rel="stylesheet" href="position.css">
    </head>
    <body>
        <div class="container">
            <button class="btn onButton" id="onButton">On</button>
            <button class="btn offButton" id="offButton">Off</butto$
        </div>
    </body>

</html>

```

A.9 position.css

```

body {
    margin: 0;
    padding: 0;
}

.container {
    text-align: center;

```

```

    margin-top: 360px;
}

.btn {
    border: 1px solid #800000;
    background: none;
    padding: 10px 20px;
    font-size: 20px;
    cursor: pointer;
    margin: 10px;
    transition: 0.8s;
    position: relative;
    overflow: hidden;
}

.onButton, .offButton {
    color: #800000
}

.onButton:hover, .offButton:hover {
    color: #fff;
}

.btn::before {
    content: "";
    position: absolute;
    left: 0;
    width: 100%;
    height: 0%;
    background: #800000;
    z-index: -1;
    transition: 0.8s;
}

.onButton::before, .offButton::before {
    top: 0;
    border-radius: 0 0 50% 50%;
}

.onButton:hover::before, .offButton:hover::before {
    height: 180%;
}

```

A.10 prawnScript.js

```
$(document).ready(function() {  
  $('#onButton').on('click', function(e) {  
    $.ajax({  
      url: '/prawnapp?status=on',  
      method: 'GET',  
      success: function(result) {  
        console.log(result);  
      }  
    });  
    e.preventDefault();  
  });  
});
```

```
$('#offButton').on('click', function(e) {  
  $.ajax({  
    url: '/prawnapp?status=off',  
    method: 'GET',  
    success: function(result) {  
      console.log(result);  
    }  
  });  
  e.preventDefault();  
});  
});
```