WORCESTER POLYTECHNIC INSTITUTE

ROBOTICS ENGINEERING PROGRAM

Lionfish Bot 2.0 2018-2019 Final Report

A Major Qualifying Project Report
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science by:

Alexander Antaya, Computer Science
Katharine Conroy, Robotic Engineering
Eric Peterson, Computer Science
Thomas Ralph, Electrical Engineering and International Studies

Project Advisors:
Professor Susan Jarvis
Professor Brad Miller
Professor Craig Putnam

Sponsored by:
Robot in Service of the Environment (RSE)

Date: 4/19/2019

Abstract

The Lionfish MQP team has worked with Robots in Service of the Environment (RSE), a subsidiary of iRobot, in order to improve their existing computer vision, electrical panel and intake mechanism systems on their lionfish capturing robot. The team created an object detection model that can identify the location of lionfish within a video, with allowances for hardware limitations, the model was then optimized for mobile computing. This model could eventually be used by the RSE team to automate the capturing of lionfish. Next, the RSE robot has two electrical panels which are used to stun the lionfish before it is captured. The team researched and tested different configurations of panels and, through this, recommended a design that would increase the effectiveness of the electrical panel system. Finally, the team developed an alternative to the existing lionfish intake mechanism on the RSE robot. The mechanism utilizes two cleated conveyor belts that work in tandem to gently ingest lionfish into the robot. We believe this research can be applied to help eradicate the invasive lionfish population in the Atlantic Ocean.

Acknowledgments:

We would like to give a special thanks to all who helped us during this MQP especially our advisors and professors, Susan Jarvis, Brad Miller, and Craig Putnam, for who we would have not been able to complete this project without. They were crucial to the completion of this project and we would have not been able to do it without their help.

Additionally, we would also like to thank Adam Cantor and the RSE team for lending us your time and resources. You provided us with vital information and for which the project could not do without. Finally, we would like to thank Professor Robert Daniello, Professor Ken Stafford, Tom Partington, and Ian Anderson for providing additional help throughout the year on this project.

Thank you!

Table of Contents

Table of Figures

1. Introduction

The lionfish (Pterois volitans) is a species native to the waters of the Indo-Pacific Ocean but over time the species has come to infiltrate the reefs of the Atlantic. It is speculated that they were introduced over the years to these waters by humans who abandoned the lionfish they had kept in home fish tanks (NOAA, 2018). While this act may seem harmless it has caused devastating effects to the reefs in the Caribbean and further north to the waters of the continental United States. These fish have venomous spines and have no naturally occurring predators in the Caribbean, meaning that there are no natural controls on their population within their new ecosystem. Without human intervention, this species will continue to grow and slowly eliminate other species of fish, impacting the entire range of diverse ecosystems in the Caribbean and Northeastern Atlantic waters.

Current efforts to mitigate these devastating effects involves monetarily incentivizing divers to collect the fish. Being that the fish have no predators it makes them relatively easy to catch as they are unafraid and slow-moving. In order to encourage divers to collect the fish, certain environmental organizations host derbies where divers compete to harvest the highest number of lionfish (REEF, 2018). Furthermore, additional incentive comes from the price at which these lionfish are sold. Although prices vary, Whole Foods Markets sells lionfish anywhere from 8-10 dollars per pound (Bowerman, 2016). This offers a fiscal incentive for divers to get out onto the Caribbean reefs and harvest lionfish. Organizations such as National Oceanic and Atmospheric Administration (NOAA) are promoting other means of lionfish harvesting by looking into innovative traps and other collection models, however, these passive solutions have limited scalability (Lieber-Kotz, 2017). While these solutions are a great idea, there are far too many lionfish for these controls to mitigate the problem and diving are often the more effective means of harvesting the fish. As such, the idea has arisen of utilizing an active autonomous robotic platform to patrol the coral reefs and fight off the invasion.

A leader in the advancement of environmental technology is Robots in Service of the Environment (RSE), which was the brainchild of the CEO and co-founder of iRobot, Colin Angle. Founded in 2015 after a dive excursion in Bermuda, Mr. Angle formed the group to develop scalable robotic solutions to environmental problems. The first was a robot that would harvest lionfish (Akpan & Ehrichs, 2016). Although still in the development stage, a goal of RSE is to create a robot that can be sold to organizations in the Caribbean to deploy and harvest the lionfish that can later be sold (RSE, 2018). Beyond the work of RSE, there are other non-profits and educational institutions developing their own take on a robotic solution. This idea of a robotic solution has not been limited to just lionfish; robots are also being leveraged to try and fight the invasive Crown of Thorns Starfish on the Great Barrier Reef (Platt, 2016). As our team seeks to continue to assist in the efforts of finding a functional robotic solution to address the infestation of lionfish, we partnered with RSE this year to assist in the development of their robotic platform.

We worked on three separate subsystems of the robot. The first subsystem is the intake mechanism of the robot. They are currently utilizing a thruster in order to suck the lionfish into the robot's capture chamber but are seeking options for an improved method of intake. The second subsystem is the electric stunning process. We will be looking to constrain and optimize the electric field employed by this system. This will include investigating the current shape of the field and how the paddle shape and other factors may influence the effectiveness and range of the field. The third subsystem is the vision and classification the robot utilizes to identify and categorize lionfish. In this portion, we will investigate how to optimize the relationship between the time it takes to classify an image and the accuracy of the identifier. Our hope is that by helping this current leader in the advancement of environmental robots we can have a lasting impact on the way people view the use of robots to help solve environmental challenges but more importantly help protect the coral reefs of the Caribbean and beyond.

2. Background

2.1 Lionfish

The lionfish (*Pterois volitans* or *Pterois miles)* is a beautiful species of fish that naturally resides in the warm waters of the Indo-Pacific Ocean (US Department of Commerce, & National Oceanic and Atmospheric Administration, 2016). Its territory includes a large span of these oceans and is found from Western Australia to South Korea. They come in many different varieties; however, they can be easily identified by their distinct scale pattern which includes white and brown or maroon stripes as seen in Figure 1. What makes this species so unique is their span of roughly thirty-three poisonous spines that cover the fish from head to tail. A mature lionfish can grow to approximately 18 inches, while the juveniles can be less than an inch.

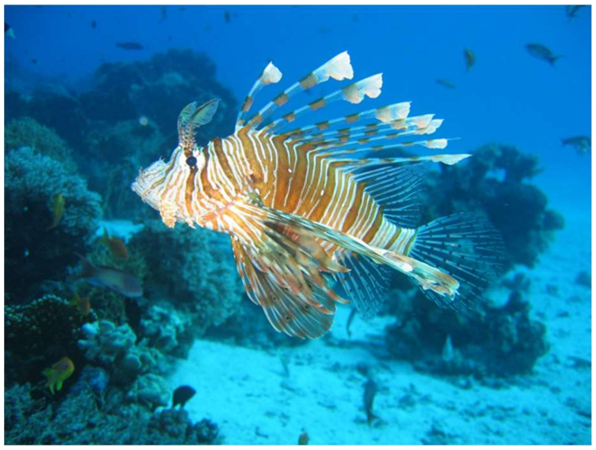


Figure 1: Beautiful Lionfish (Pterois volitans) swimming. (Skeeze, 2015)

2.1.1 Invasive Species

Scientists theorize that lionfish were first introduced into the Caribbean region in 1985. Since their introduction, their population has grown rapidly (US Department of Commerce, &

National Oceanic and Atmospheric Administration, 2004). In 2002, they were seen off the coast of North Carolina and since then they have been seen in regions as far North as Long Island, NY and Cape Cod, MA. Contrary to popular belief, hurricanes (Morell, 2017) are not the cause of the current lionfish problem; it is more likely due to aquarium owners who have dumped their unwanted lionfish into the ocean. These fish have done unusually well as a species in adapting to these cooler waters and have caused many environmental problems. One of the difficulties with the species is that there are no natural predators in the Atlantic Ocean to keep them in check. To make matters worse, lionfish breed year-round in the Caribbean releasing eggs every 4 days. In a single year, a mature lionfish can produce up to two million eggs (National Geographic, 2016). Along the east coast and in the Caribbean waters, lionfish mature faster than the native fish species and eat almost anything they can fit in their mouths. Lionfish's high reproduction rates and bottomless appetite make them an apex predator around reefs in the Atlantic Ocean. Additionally, in the Caribbean, they are consuming many of the fish that maintain the coastal reefs, which is causing tremendous damage on the already stressed ecosystem (Green, Akins, Maljković, Côté, 2012).

2.1.2 Mitigation of Lionfish

Many organizations such as the Reef Environmental Education Foundation (REEF) is working to mitigate this problem through educational efforts and derbies. In conjunction with Whole Foods Market, they host a series of fishing derbies in which teams are tasked with collecting as many lionfish as possible between sunrise and 5 pm. They are then awarded a variety of prizes in several categories (REEF, 2018). REEF believes these derbies help in more ways than just decreasing the population. They see it as an opportunity to educate the public on the issue, train the divers on how to harvest the fish and create a market for these fish as food. According to

REEF’s website, they have collected 23,332 lionfish in these derbies since 2009. While this number is significant and helps to address the problem, that number is very small relative to the total population of lionfish currently in the Caribbean.

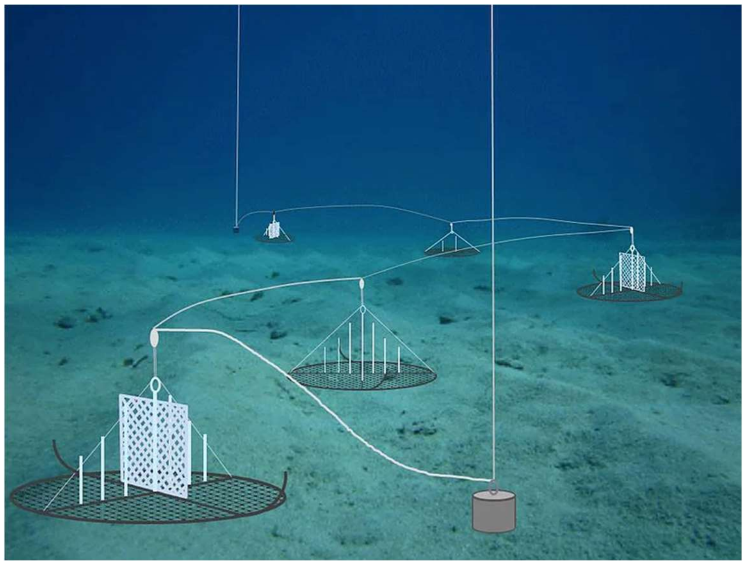


Figure 2: Innovative lionfish trap that act as a removable coral reef (Lionfish University, 2018)

Other organizations are focusing more of their efforts on developing means by which to passively trap the fish. Lionfish University, another non-profit, is working with NOAA to develop an effective trap, depicted in Figure 2, designed to catch lionfish (Lionfish University, 2018). These traps are used as opposed to divers due their ability to go to the depths at which large groups of lionfish can be found, which is far beyond the ability of human divers. Their current design utilizes a large circular net on a frame that rests on the ocean floor. In the middle of the trap is a pop-up structure that looks like a reef structure which attracts lionfish. When the strap is pulled up, the circle folds in half and the net is thus closed onto itself to form a semi-circle. The issue with this design, however, is that the lionfish are only entrapped if they are within the circle when it is being pulled up. As such, if there are lionfish in the trap at one point in time, they may leave before the trap is pulled. In this sense, the percentage of fish the trap can catch may not be enough in comparison to other methods.

New trapping methods are not the only way organizations are dealing with aquatic invasive species. At the Queensland University of Technology (QUT), scientists have developed a submersible robot capable of navigating reefs, identifying Crown of Thorn Starfish, and injecting them with poison. This platform shows that robotic systems can be a viable solution to helping maintain the health of reef systems.

While both REEF's and Lionfish University's efforts are great examples of non-profits doing valuable work to help combat lionfish in the Caribbean, we believe utilizing a robotic solution will be a valuable tool in the fight to limit their impact on the Atlantic ecosystems. In conjunction with education, traps, derbies and recreational divers, a robot would add an active deployable solution that may be more effective for entities to utilize versus using human divers.

2.2 Work Done by Prior MQP

The 2017-2018 Lionfish Bot MQP team sought to engineer a way to solve the lionfish epidemic plaguing the waters of the Caribbean and Eastern United States by creating a robot to autonomously find and catch these fish (Godsey, Kelly, Lombardi, Uvarov, Yuzvik, 2018). During the beginning of their MQP, the team researched the numerous ways of capturing lionfish to gain inspiration for the device. Additionally, they studied the restrictions of lionfish hunting along with the environmental concerns that they would need to keep in mind such as the already endangered coral reefs. A list of specifications was created to ensure that all goals were met while keeping in mind the concerns of the reef and the Caribbean governments. This list included the requirements of autonomy, accuracy, neutral buoyancy, harvesting efficiency, target discrimination, reliability, and safety (Godsey, Kelly, Lombardi, Uvarov, Yuzvik, 2018).

Due to limited funding and ease of use, the team decided that the best plan of action was to design an attachment for a commercially available ROV. The BlueROV2 was chosen by the team due to its open-source software, mechanical reparability, overall movability, payload capacity, sensor operability, and future improvement potential. This attachment for the BlueROV2 would be a proof of concept for when the team was able to receive the ROV. Once it was received, integration would be straightforward.

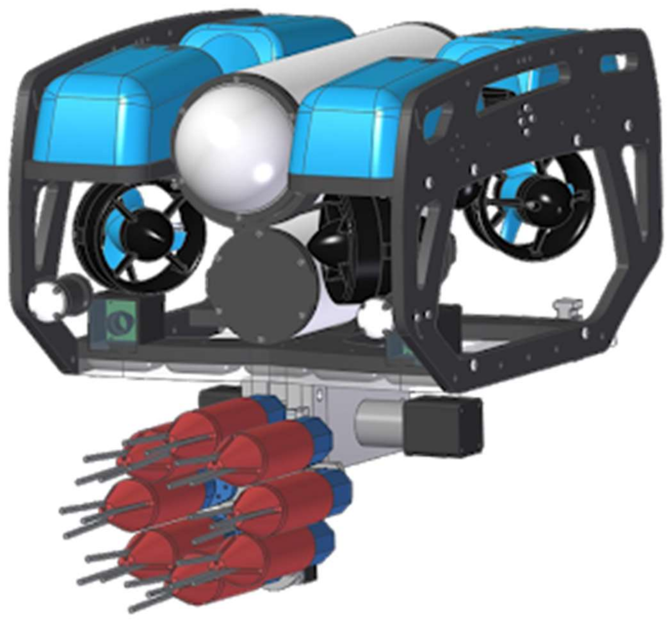


Figure 3: Prior MQP Attachment

The attachment design, depicted above in Figure 3, included many innovative aspects such as computer vision to autonomously find lionfish in the field of vision and drawing a delineating box. This was a multistep process that utilized the open source program, TensorFlow, to develop a neural network for recognizing lionfish with a high success rate. This neural network was also trained to identify non-lionfish objects for safety. Once a lionfish was recognized, a bounding box was drawn around the lionfish and the coordinates of the target were then sent back to the ROV.

They learned that computing this data on the Raspberry PI 3 was very slow with runtimes of 4-5 seconds per frame, far slower than what was required to generate steering commands in real-time. In an effort to improve the computational power of the system, the team decided to use an Intel Movidius™ Neural Compute Stick for the process and improved the runtime to 250 milliseconds per frame.

Mechanically, the attachment needed to be efficient, reliable and safe. The prior team implemented a Geneva Mechanism to implement a rotating spear device. The Geneva Mechanism allowed for the actuation of eight spear tip into place. After each positively buoyant spear tip was dispensed, the Geneva Mechanism rotated until the pin of the rotating drive wheel is caught by a driven wheel. This driven wheel was subsequently placed in the correct position to eject another spear tip. The spearing mechanism was similar to a current lionfish hunting mechanism that propels the spear into the fish and then retracts for the next spear tip (Godsey, Kelly, Lombardi, Uvarov, Yuzvik, 2018).

At the conclusion of their project, the team had created a fully functional attachment capable to identify and spear lionfish. However, they were not able to gain the funding to purchase the BlueROV2. As a result, the integration of the attachment was never accomplished leaving the acquisition of a an ROV platform and autonomous navigation of the robot as future work.

2.3 Robots in Service of the Environment

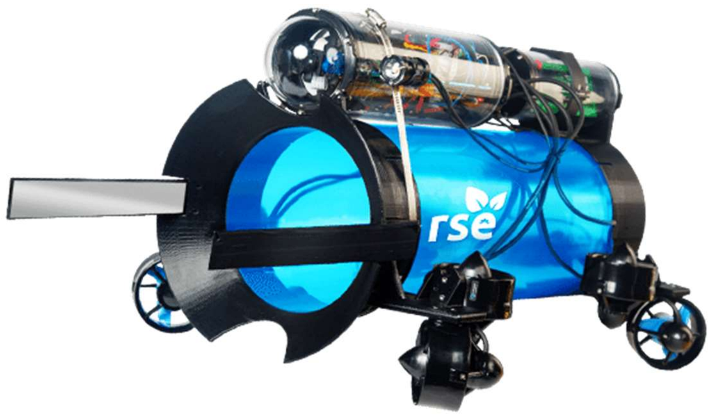


Figure 4: RSE Robot (RSE, 2018)

The Robots in Service of the Environment (RSE) robot, depicted above in Figure 4, is designed to be a non-contact, non-lethal capture and storage system. The body of the robot is a cylindrical tube that can hold more than 10 lionfish. The platform has two metal panels attached to the front of the robot to deliver a stunning shock to the lionfish. Another system on the robot then brings the stunned lionfish into the storage chamber. The entire project has been built with open source and easily available affordable parts that allow the platform to be very accessible and affordable to users. The platform was originally tested in the spring of 2017 when the team took their system to Bermuda and demonstrated its capabilities. Since then, the project has developed in a variety of ways and has undergone several changes to its structure and system, including a complete overhaul of the electrical systems, modular separation of subsystems for ease of work and several changes to the suction system used for capture. The team is currently on their third prototype and will be moving to their fourth and final prototype in the near future. The new prototype will offer several changes to the structural design of the robot and make it significantly more durable. This is the model the team plans to perfect and market by the beginning of 2019.

Long-term plans for the robot include a second version which will feature autonomous functionality which will allow the robot to operate independently of a driver.

The current Lionfish Team is an entirely volunteer-based group of 26 individuals, who are also mostly employees of iRobot. Due to the volunteer nature of the team, funding for the RSE platform is limited and time dedicated to the project is not always consistent. While our team was researching funding options for our original system plans, we recognized the opportunity to gather information from iRobot. These meetings led us to believe that there were design challenges facing the current RSE design team, with which we could possibly be of assistance. After speaking with the head of the RSE design team, we were presented with several things we could do to help the effectiveness of the RSE robot in the water, as well as, complete the requirements of our MQP. These tasks include designing and optimizing the electric field stunning mechanism, the capture mechanism and refining the computer vision system. Collectively these three smaller projects will give us the opportunity to complete our MQP and be of assistance to the RSE team in accomplishing our mutual goal of using technology to tackle the lionfish problem in the Caribbean.

2.4 Classification System

A critical capability that the system must have is the ability to autonomously detect the presence of a lionfish in the robot's environment. The concept of a program gathering information about its environment, especially image data, and reacting in a predefined and predictable manner is known as Computer Vision. A subset of computer vision is image classification, where a computer is 'taught' what a specific object looks like and can classify subsequent images with a certain level of confidence. There are three major factors associated with image classification: image classifier architecture, the dataset used to teach the image classifier, and the testing the image classifier to determine appropriate thresholds for discrete classifications.

2.4.1 Image Classification Techniques

There are several different techniques that researchers have developed to classify images. However, deep learning has emerged as the superior method for image classification. Deep learning utilizes artificial neural networks to emulate the way the human brain learns, processes, and labels visual data (TensorFlow, 2018). There are also many advantages to using neural networks such as the ability to parallelize computation and reuse the same general architecture to solve many different image classification problems (Stanford, 2017).

Artificial Neural Networks (ANNs) work by breaking down a problem into smaller sub problems and identifying patterns. It then uses this information to construct a decision graph with weighted neurons (nodes in the graph) which is subsequently executed using new data to return a probability that the image fits a certain label based on the input data (Raj, 2018). Convolution Neural Networks (CNNs) are a specific architecture of ANNs which has at least one convolutional layer. Specifically, CNNs have proven to be the most effective at classifying images, and as such, they are the most common architecture for image classifiers. In image classification, the CNN works by taking an image and breaking it down into subgroups of pixels. It then takes those pixels to create lines and shapes, then it will take the shapes that it found and associate those to features, and with those features the CNN can make a prediction of what is in the image.

Take this image of a dog for example, which can be seen in Figure 5. A dog can be broken down into smaller features such a head, four legs, and a tail. The CNN would then break down a specific feature into lines and shapes such as a leg which is depicted in Figure 6.

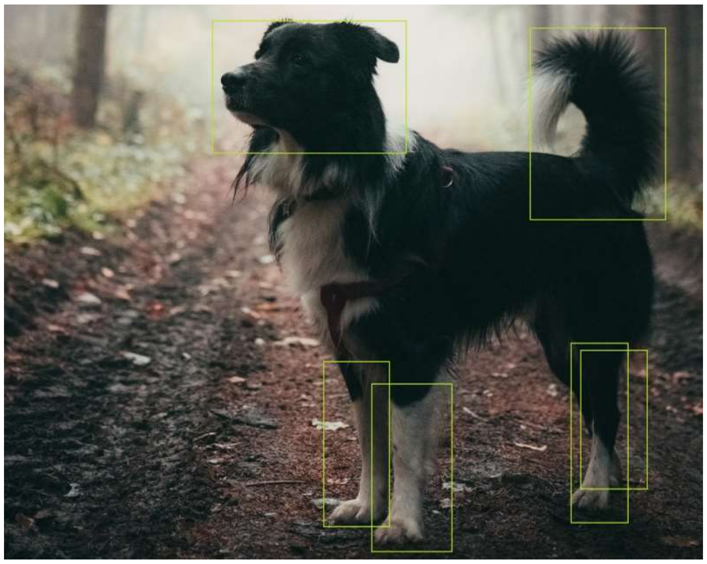


Figure 5: Dog Annotation

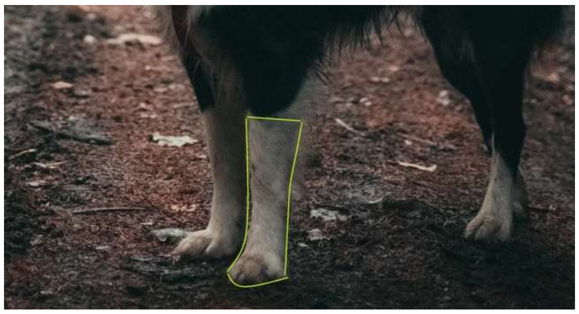


Figure 6: Dog Leg Annotation

In reality, the CNN works in reverse of this, where it will build up from small groups of pixels into lines and shapes into features then into groups of features, and so on, until it has progressed through all of its 'layers' and can make a prediction as to what the image is depicting (Stanford, 2017).

There are two ways to create a CNN; one could create an CNN from scratch, or one could retrain an existing CNN. Creating a CNN from scratch requires a fundamental understanding of machine learning, CNNs, and a very large dataset (TensorFlow, 2018). Developing a CNN from scratch also increases development time since researchers will need to find much more data, annotate it, develop the code for generating a CNN, train it on the large dataset, and test it. After all this, researchers will still most likely have to adjust parameters such as the number of layers, the types of layers, the number of epochs for training, and numerous other parameters which all effect the CNN's accuracy and efficiency. The alternative is to retrain an existing CNN. This works by taking an existing image classifier and retraining it to classify new images. The researchers will still need to find and annotate a dataset to train the image classifier, however, it will not need to be nearly as large as the dataset required to train a CNN from scratch. The reason why retraining does not require as large of a dataset is because retraining preserves the feature extracting portion of the image classifier and simply retrains a new classification layer on top. One downside to retraining, also known as transfer learning, is that the model will not be quite as accurate as developing a model from scratch (TensorFlow, 2018). This is because the model that was developed from scratch will most likely be able to perform feature extraction better since it was trained only on the specific features of the desired classifications. However, for most applications retraining an image classifier will provide enough accuracy.

2.4.2 Choosing an Image Classifier

There are several factors to consider when choosing an image classifier to retrain. One of which is accuracy. Using an image classifier with a large and complex model will generally provide better accuracy than a smaller or 'slimmer' model. Also, one must consider speed. Although a larger model will provide more accurate results, it will also take longer to calculate those results. Furthermore, a smaller model will also consume less memory and disk space, both of which are a concern when developing a solution for a mobile platform. For some applications, especially mobile ones, a less accurate but faster image classifier is preferred (D'Almeida, 2018).

One commonly used image classifier is the ‘Inception V3’ model. Inception is a model with high accuracy results and moderate running time. The model was trained on the ImageNet dataset which is a vast collection of over fourteen million images gathered specifically for training and testing image classifiers (TensorFlow, 2018).

Another common class of image classifier is MobileNet. MobileNet is a light weight deep neural network that, as the name alludes to, has been developed specifically for mobile and embedded applications. MobileNet introduces two simple global hyper-parameters that efficiently tradeoff between latency and accuracy. This model has also been tested against the ImageNet dataset (Howard, et al., 2017).

Figure 7 and 8 below represent the relationships between model size, accuracy, and latency/speed. The Figure 7 shows that the Inception models have the highest level of accuracy but come at the cost of disk space and memory consumption. Figure 8 shows that as a model increases in size, so does their latency (TensorFlow, 2018). The models that have been optimized for mobile development, are smaller in size and provide significantly lower latency.

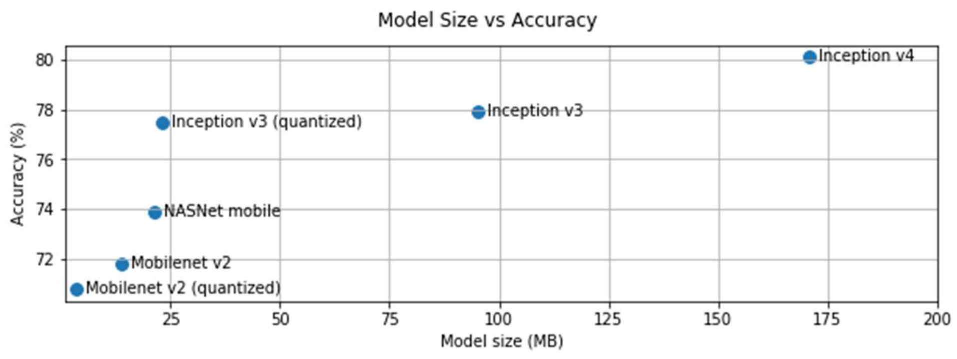


Figure 7: CNN Model size vs Accuracy (TensorFlow, 2018)

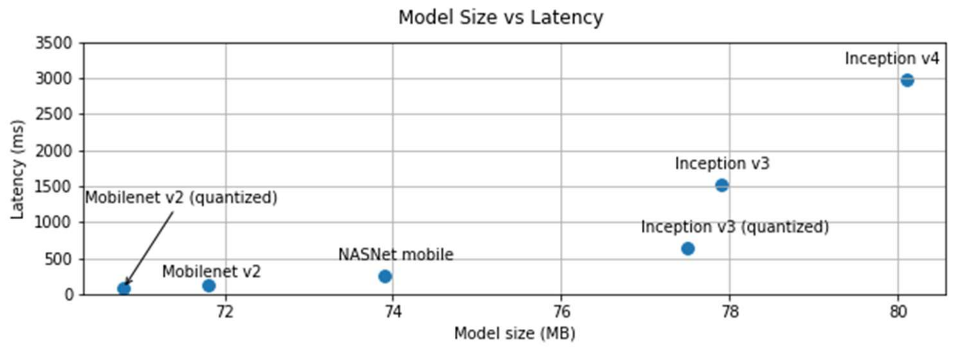


Figure 8: CNN Model Size vs Latency (TensorFlow, 2018)

2.4.3 Creating a Dataset

Developing an appropriate and thorough dataset is an important aspect of creating an accurate and robust image classifier. The dataset can directly influence properties of the image classifier such as invariance. Invariance is simply the image classifier's ability to accurately identify images independently of the image's view point, scale, deformation, occlusion, background clutter, illumination, and intra-class variation. Figure 9 represents examples of what these properties may look like in real-world applications.

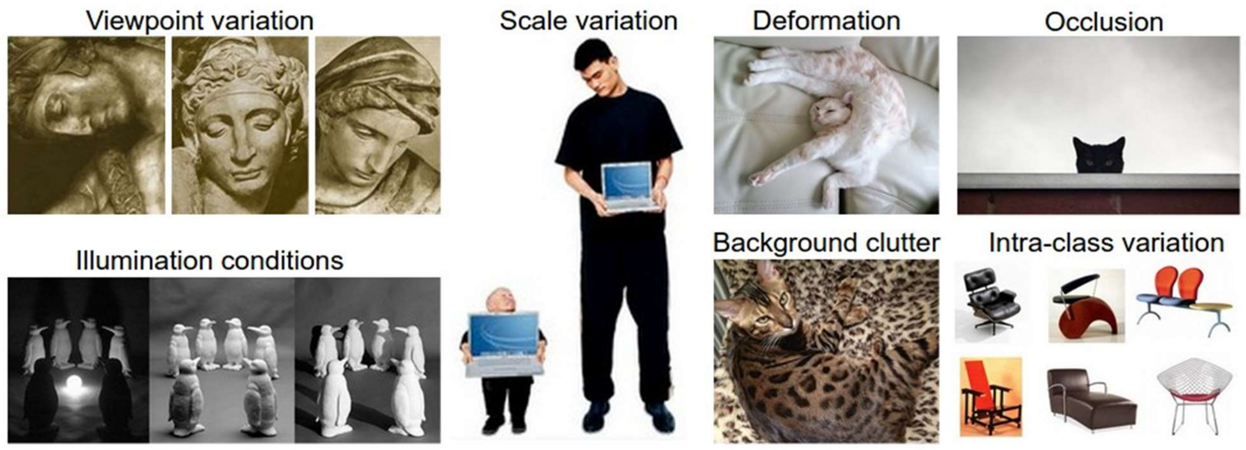
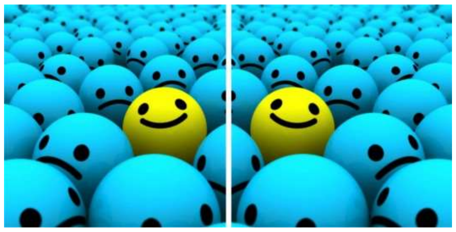


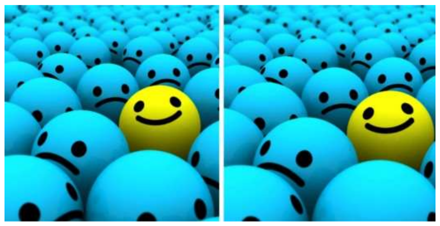
Figure 9: Image Characteristics that affect how an image is viewed (Stanford University, 2017)

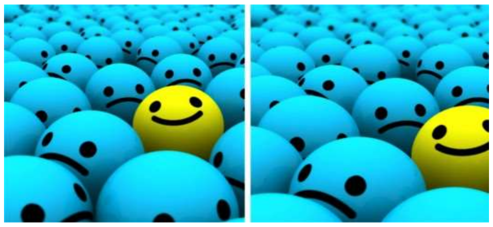
Illumination conditions, occlusion, viewpoint variation, and background clutter are all properties that are very common to see in real word applications. The effects of illumination conditions are very drastic at the pixel level. Occlusion is when only a portion of the object is present in the image. View point variation is one of the most important, because it will practically always come into play in real world applications and must be accounted for in the dataset (Stanford University, 2017).

Another problem with image classifiers is that in order for them to be accurate they must be trained or retrained on a relatively large dataset, usually on the order of thousands of images (TensorFlow, 2018). Finding a large enough dataset is something that many researchers find challenging. There are several ways that researchers can create a dataset. They can find a pre-exiting dataset, make their own using images found online, they can make their own dataset using their own images, or a combination of the three.

Data augmentation is a technique used by researchers to solve the problem of invariance and small datasets (Raj, 2018). Augmenting a dataset basically means, taking a pre-existing dataset and modifying the images slightly. There are several operations that are common amongst data augmentation programs such as flipping, rotating, scaling, cropping, and translating. Figure 10 depicts examples of what flipped, scaled, cropped, and translated images would like.


Flipping an image


Scaling an image


Cropping an image

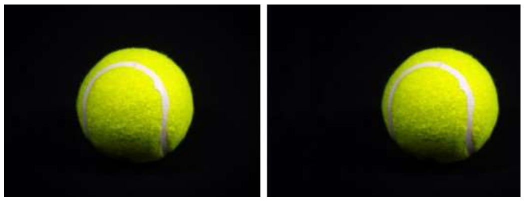

Translating an image

Figure 10: Image Operations that effect how an image can be viewed

The reason why data augmentation works is because during the training process the image classifier does not actually understand that the two images in Figure 10 are actually the same image; to the image classifier they are both unique images. What that means is that, within reason, we can perform variations of these operations on all the images in our dataset and increase our dataset size. For example, if we took the dataset and just performed a translation operation four times for all the images in our dataset, we would increase the size of our dataset by a factor of four. However, data augmentation can have negative effects on the dataset if done improperly. Take Figure 11 for example, this image is being flipped in an orientation that the camera would most likely never capture in most real-world applications (Raj, 2018).

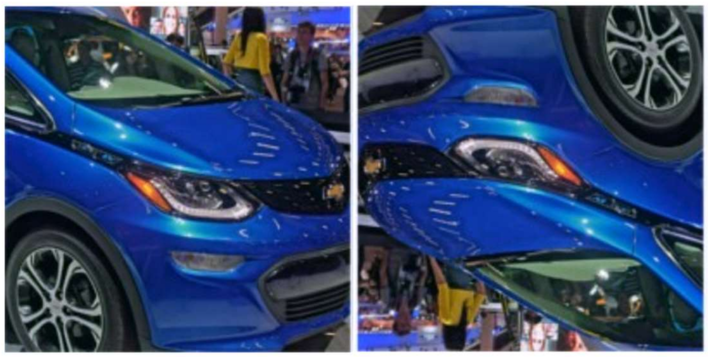


Figure 11: Image Flip (Raj, 2018)

2.4.4 Testing Image Classifiers

The performance of these image classifiers can be measured in a variety of ways and are impacted by numerous factors. As we seek to test the classifier, how we will be creating it will be important to understand how to go about testing it and the factors to be considered.

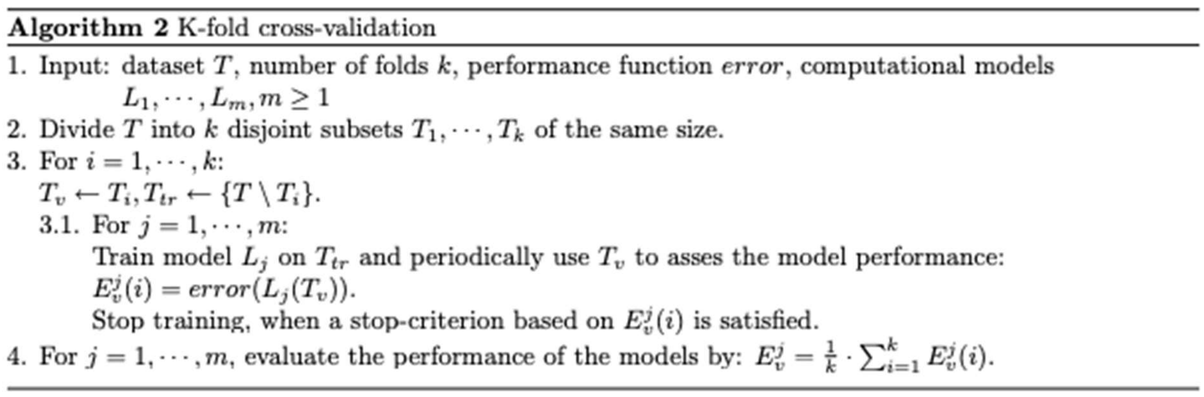
2.4.4.1 Training Set vs. Test Set

Once a dataset has been created and the CNN has been retrained, testing must be done in order to gain an understanding of the performance and efficiency with a variety of factors to be considered. Some of these methods include Holdouts, random subsampling, cross-validation, and stratified cross-validation.

The Holdout method of testing an image classifier is a fairly simple one. When providing the CNN with the dataset, a portion of that dataset is to be "held out". This portion which is held out is then utilized as the test dataset, leaving two sets of data: training and testing. An arbitrary breakdown for this separation is 2/3 of the data used for training and another 1/3 used for testing (Kohavi, 1995). This ratio however has its tradeoffs. With a smaller training set the classifier may not be as accurate and will have a higher bias. But if you don't have a large enough test set the confidence you have in the accuracy of the classifier goes down because it hasn't been run on as much data.

In order to try and mitigate this problem a random subsampling method can be utilized. In this method, holdout is repeated multiple times in which a random 1/3 (or other determined portion) of the total dataset is picked out for testing each time (Kohavi, 1995). By then averaging the accuracies of these runs we can better understand the total performance of the classifier. However, the issue here is that not all the data in the set is totally independent and a certain bias could be created if the variance between the images being used to train the CNN is not enough and the same could be said for the test set. In general, the difficulty with holding out portions of data is it makes inefficient use of the data because it is not fully training the classifier with all of the data that could potentially be used.

Another method is cross-validation; in this method, you randomly divide the dataset into a certain number of equally sized subsets or folds. The CNN is then trained and tested for however many folds you have, leaving out a single fold each time (Reitermanov´, 2010). Then in order to get a performance result based on that you divide the number of correct classifications by the number of total instances of that class. This gives an overall average for the test results over the total number of folds. The general algorithm for this method can be found below in Figure 12. The downfalls with this method are still the incompleteness of the testing. It would be too much to go through large datasets and test every combination of folds, just as it was for holdouts (Kohavi, 1995).



```
Algorithm 2 K-fold cross-validation
1. Input: dataset $T$, number of folds $k$, performance function $error$, computational models
          $L_1, \cdots, L_m, m \geq 1$
2. Divide $T$ into $k$ disjoint subsets $T_1, \cdots, T_k$ of the same size.
3. For $i = 1, \cdots, k$:
   $T_v \leftarrow T_i, T_{tr} \leftarrow \{T \setminus T_i\}$.
   3.1. For $j = 1, \cdots, m$:
        Train model $L_j$ on $T_{tr}$ and periodically use $T_v$ to asses the model performance:
        $E_v^j(i) = error(L_j(T_v))$.
        Stop training, when a stop-criterion based on $E_v^j(i)$ is satisfied.
4. For $j = 1, \cdots, m$, evaluate the performance of the models by: $E_v^j = \frac{1}{k} \cdot \sum_{i=1}^{k} E_v^j(i)$.
```

Figure 12: K-fold cross-validation algorithm (Reitermanov´, 2010)

Just as random sampling improved the accuracy of holdout testing, stratified cross-validation is similar. In this method, you once again utilize an arbitrary number of folds for the testing and those folds are tested the same way as shown in Figure 12. However, the difference is that those folds are stratified so that they each contain approximately the same proportions of a specific class or label (Kohavi, 1995). This uniform distribution helps to limit the bias that could be created in the classifier if the training set contains much more of one class then another based on the random separation into folds. The difficulty with this method is choosing the right parameters by which to separate out the clusters that will be pulled from into each fold (Reitermanov´, 2010). Every one of these options for training and testing have some sort of drawback and the decision to pick one depends heavily on the size, diversity, and type of data set being used.

While the above methods are effective baselines for evaluating an image labeler, there are different evaluation techniques that can be considered for object detection models. Specifically, these methods include: Precision, Recall, and Mean Average Precision(mAP). Preicsion and Recall are both calculated utilizing a certain decision threshold and the values of both will vary depending on what that threshold is. Precision is defined as the ratio of correct positive classifications to the total number of positive classifications made by the model, this value increases as decision threshold increases. This gives an idea of how accurate the model is being when it tells you there is a certain class present in the frame or image. Recall is the ratio of positive classifications to the number that should be classified. This defines, in this case, the percentage of lionfish the model actually identifies, decreasing as the decision threshold increases.

The formulas for calculating these values utilize the terms true positives, false positives, true negatives, and false negatives. True positives are when the model identifies an object and it is in fact that object. False positives are when the model identifies an object but that object isn't actually there. True negatives are when the model doesn't identify the object and it isn't actually there. Lastly, False negatives are when the model doesn't identify the object but it is fact there. The formula then for precision is true positives / (true positives + false positives). The formula for recall is true positives / (true positives + false negatives). The meaning of these formulas are described more in the preceding paragraph.

Utilizing mAP is a different technique. mAP is averaging the recall and precision values across all the positive identification thresholds, i.e. by setting a model to accept if 20 percent confident or if 90 percent confident (Arlen, 2018). Then these points can be graphed on a recall vs precision graph. Then to calculate the mAP we average the precisions across all recall values in increments of 0.1 from 0 to 1.0. This helps to define the ability of the model to classify but still doesn't help us understand the ability of the model to localize the position of the object in a frame. To do this we utilize an intersection of union value as a filter for positive identifications. This intersection of union takes the percentage of the bounding box created by the human and that created by the model and returns the percentage by which they overlap. With this added filter the mAP value provides an accuracy figure representing classification and localization of that object in the frame.

2.4.4.2 Image Quality

One of the constraints in testing a classification system is the quality of the images being provided by the camera underwater. Ideally, we would use the same camera for training dataset collection and actual classification. However, this option was not entirely feasible. To overcome this challenge an option was to enhance images utilizing software to meet a specific standard before being passed to the CNN for classification. Specifically, for the application of RSE, image enhancement can be especially challenging because a portion of the work can be done in naturally lit water versus the consistency that comes with an on-board light and camera (M. Roser et al., 2014).

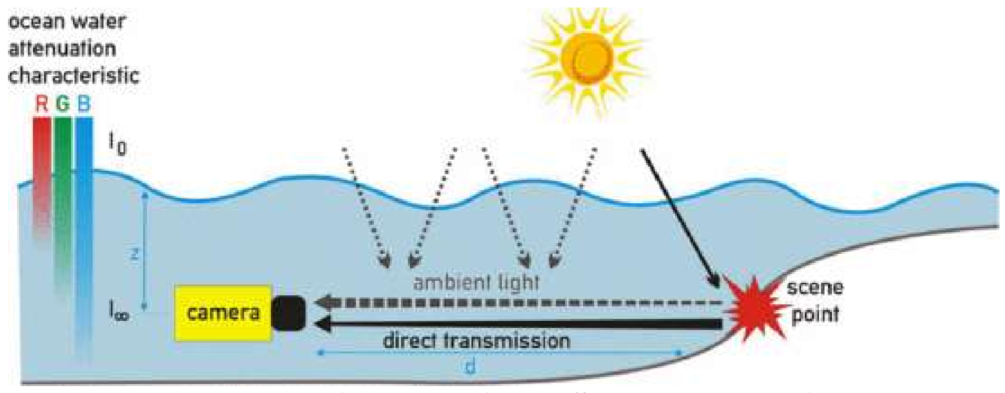


Figure 13: Underwater Lighting Effects (M. Roser et al., 2014)

Some of the factors that come into play when gathering image data underwater can be seen in Figure 13 above. With these factors in mind, one is able to make some calculations regarding the distortion of images with factors including backscattering, forward scattering, floating particles in the water, and light absorption (AbuNaser et al., 2015). The methods to overcome these challenges involve complex calculations and almost endless factors to consider. This means that the number of articles and approaches to image enhancement are broad and varying in thought. The most effective methods use a range of other algorithms in a step-by-step process. The varying results of these methods are referenced in Figure 14 below which shows method's Quality Index based on water turbidity and comes from a paper on a proposed *"novel joint guidance image de-scattering and physical spectral characteristics-based color correction method"* (Li et al., 2016).

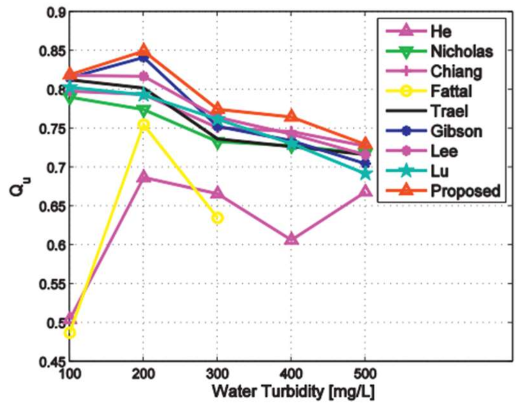


Figure 14: Image enhancement methods (Li et al., 2016)

These varying methods provide valid options for possible additions to the classification system to ensure it works effectively. The image quality will certainly be a factor when testing any underwater classifier and the proper image enhancer to utilize will vary based on image types, operation depths, and computational power necessary.

2.4.4.3 Software Options

When testing the effectiveness of an image classifier using a CNN it is important to get numbers on accuracy as discussed in section 2.4.4.1. However, another consideration that comes into play is the software being utilized to create the model that the CNN will be trained with. More popular options for this model creation and implementation include TensorFlow and TensorFlow Lite. These two methods provide varying results in terms of computational resources required and time to accurately classify.

TensorFlow is an open source library developed by Google, Inc. for a wide variety of machine learning capabilities (TensorFlow, 2018). In our case, this library can be used in order to easily retrain the inception layer of the CNN. The library itself is at the forefront of machine learning development, making it easy to develop with. Some of the limitations with TensorFlow, however, is that it will only work on a 64-bit machine and it is not supported for the latest version of Python (3.7). Additionally, it is used to develop ML on a very broad scale including clusters of CPU and GPUs, desktop, mobile devices, cloud, and web (TensorFlow, 2018). The downfall of this capability is that it is not optimized for any specific device type.

In 2017, TensorFlow addressed this problem and released a new platform for ML development, TensorFlow Lite (ICT Monitor Worldwide, 2017). This library streamlines the capabilities of TensorFlow for specific use on mobile and embedded devices. Some of the features that TensorFlow Lite boasts is increased speed with no noticeable accuracy loss, low latency, smaller model sizes and better tooling (TensorFlow, 2018). Additionally, TF Lite offers a simple convert tool in order to convert a TF model to the Lite version. The converter works by taking the saved TF model, turning that into a FlatBuffer file, that file is then deployed to the mobile device and the compressed model is used by the TFLite interpreter, as shown in Figure 15 below (TensorFlow, 2018). In testing, it will be important to look at the difference in performance of the two options and determine which platform will be most effective for our specific application.

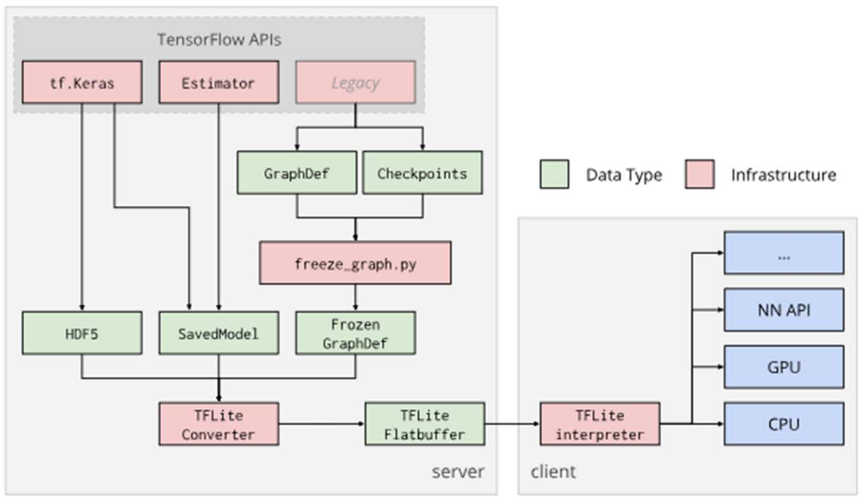


Figure 15: TensorFlow Lite Converter Flow (TensorFlow, 2018)

2.4.5 Hardware Options

In an ideal world, we would utilize the exact same hardware being utilized on the current RSE platform. However, the board and camera module are proprietary, and we will have to instead choose a different baseline hardware configuration as a proof of concept. In conversations with our contact at RSE and based on the work of the 2017-2018 MQP team we will be utilizing a Raspberry Pi 3 Model B and Raspberry Pi camera module V2. This should provide a baseline configuration for testing to be run on our classifier.

Figure 16: Raspberry Pi3 Model B (Raspberry Pi, 2018)

The Raspberry Pi 3, depicted in Figure 16, provides a single board computer on which we can download a Light version of Linux operating system in order to run the CNN we decide to utilize. The board comes equipped with a Quad Core 1.2GHz Broadcom 64bit CPU, 1 GB RAM, a CSI camera port as well as numerous other functionalities. In the case of image processing, a resource-intensive process, the Raspberry Pi may not be able to provide real-time inference. With this in mind, it will still provide a performance baseline to test differing software configuration.

Figure 17: Movidius Neural Compute Stick (Intel, 2018)

The simple fix to speeding up the image inferencing process on a Raspberry Pi is by utilizing an Intel Movidius™ Neural Compute Stick (MNCS), depicted above in Figure 17. This device allows for the deployment of ML models to be implemented on-device with acceptable performance for real-time inference (Intel, 2018). Movidius provides a simple SDK in order to convert an already trained model into a compiled model to be run on the host, in this case, the Raspberry Pi 3 Model B (Kizrak, 2018). A visual representation of utilizing MNCS can be found in Figure 18.

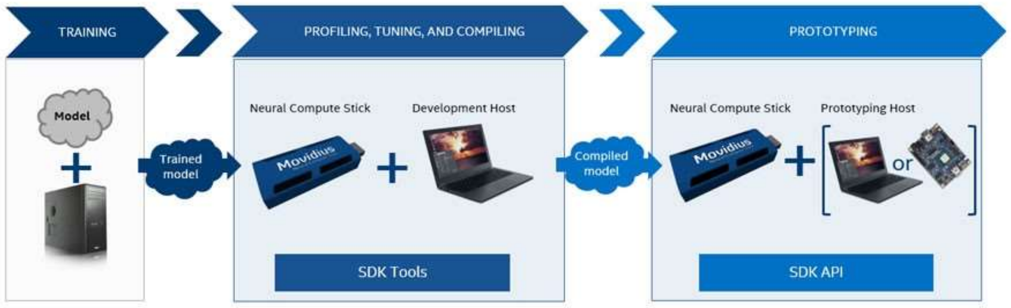


Figure 18: Compute Stick Operation (Kizrak, 2018)

While the MNCS is likely to be an effective solution to develop a real-time image classifier, it does not meet the needs of RSE. Being a company with a product they will obviously be looking for a cost-effective solution to the problem and the cost per unit of the MNCS does not currently meet that need. However, it will provide a useful option to utilize when testing different model and software configurations for real-time classification.

2.5 Electrode System

The RSE platform implements a stunning mechanism that temporarily paralyzes a lionfish. It does this by generating an electric field between two conductive plates, which are placed in front of the capture chamber. The person controlling the robot navigates until a lionfish is between the two plates and then delivers a shock to the fish. Once stunned, the fish is pulled into the capture chamber. The design of the shock system is very simple; the two plates are attached to plastic runners that extend out from the frame of the robot. The system is powered by a 28V source and the specifications of the shock have been customized by the RSE team to effectively shock and temporarily paralyze the species of lionfish present in the Caribbean.

After speaking with the RSE team, we identified the electric shocking system as one of the areas of the platform that could use improvement. Most notably, after several field tests in the Caribbean, the team has determined that the current field configuration requires a lionfish to be within approximately 3 inches of either plate in order to administer the shock. The RSE team requested that our team document the shape and properties of the field, explore methods of improving the shocking consistency of the system and explore the possibility of projecting the field forward to shock lionfish in front of the panels.

2.5.1 Electrofishing

In principle, the RSE system operates similarly to the practice of electrofishing. Electrofishing uses a system composed of anodes and cathodes to generate an electric field that attracts fish toward the system. As the fish approach enter the electric field, they are involuntarily drawn toward the anode in the biological process known as galvanotaxis. The high voltage anodes create a current between them and the cathode, which is often the metal boat itself. When the fish reaches the anode and passes through the current, it enters narcosis, which renders it temporarily paralyzed for a short period of time (Sparks, 2003). When the fish, already near the surface is stunned, it floats upward and is captured by the worker on the boat. During this process, a delicate balance between pulse and current must be maintained in order to effectively capture fish without killing them. This practice of capturing fish is best done in shallow areas and works best on territorial fish that are not easily scared. ("ElectroFishing, 2009").

The process has two distinct steps, using a field to attract fish and using the current generated by the field to shock the fish. While there are similarities to the system used by the RSE team, generally, electrofishing is used to attract fish, without discrimination of fish type. The RSE platform is specifically looking to harvest a single type of fish. Further, the United States Fish and Wildlife Service defines electrofishing as "using electricity in aquatic habitats to sample or control fish" (Occupational Safety and Health, 2016). By this definition, the system utilized by the RSE team is a unique type of electrofishing. Electrofishing is also largely restricted to freshwater, specifically rivers and ponds, due to the limited depth of the process.

2.5.2 Field Properties and Shape

The properties of electric fields form the basis of our project. An electric field is the force per electrical unit charge. The field extends orthogonally out from a positive charge and in toward a negative charge. The field for an electric dipole is shown in Figure 19. In this example, the field can be seen radiating away from the positive charge and toward the negative charge. As the field radiates toward or away from a charge, it creates equipotential lines that represent electric potentials.

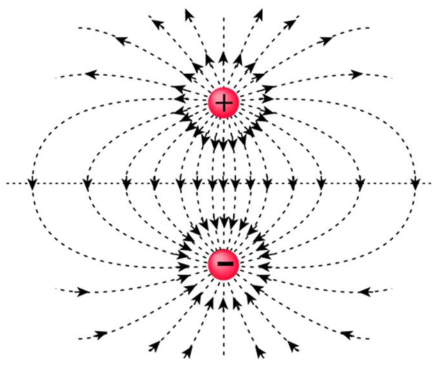


Figure 19: Electric Dipole Field

The RSE electrode system represents a larger scale version of an electric dipole. One panel is charged to a positive voltage; the other panel is grounded (negative charge). As a result, a similarly shaped electric field is generated by the system. Figures 30 shows a scaled model simulation of the RSE system. In these simulations we can see that panels create a field shape similar to the field created by the electric dipole. Figure 20 also shows the field strength produced by the simulations. As we can see, the field is strongest near the panels, however, the strength narrows near the center of the panels.

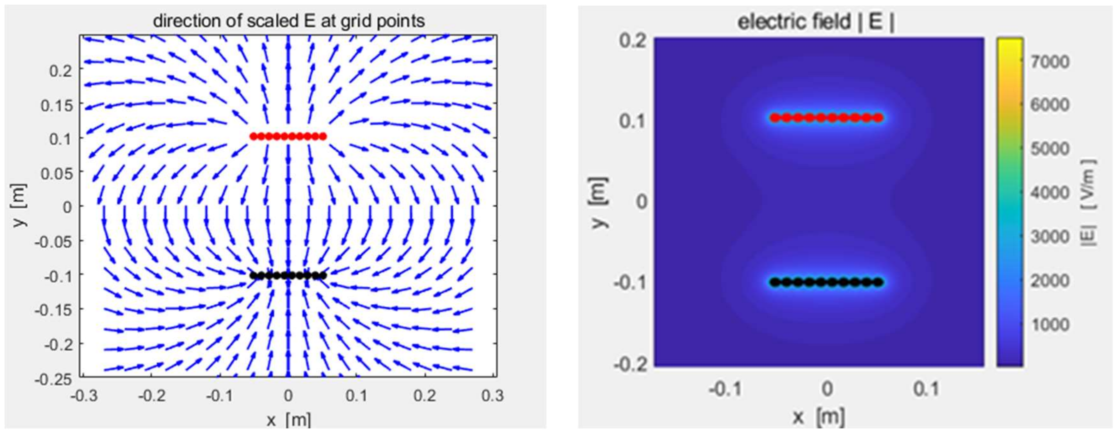


Figure 20: Simulated Electric Field (Left) and Field Strength (Right)

2.5.3 Cassini Oval Effect

Our research of electric fields led us to the discovery of an interesting physics theory. A Cassini Oval is defined as a curve around two points, such that the product of the distance between the two points remains constant. A three-dimensional model of a Cassini Oval is shown in Figure 21. A real-world example of this principle is most commonly seen in the shape of peanuts. The idea of a "peanut" shape field was part of the initial information given to us by the RSE team. This shape was also similar to the field strength lines we saw during our MATLAB simulations.

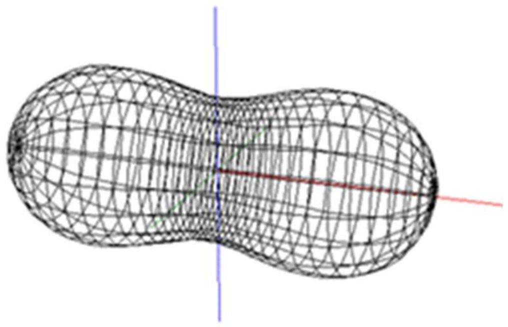


Figure 21: Cassini Oval Model

Given the similarities between the shape of the field strength lines and the predicted "peanut" shape offered by the RSE team, the Cassini Oval was a theory we needed to explore. The issues with this model, however, work on the assumption that the electric field itself delivers a shock to the fish, when it is the current travelling between the two panels that actually shocks a fish. Regardless of the field strength outside the area directly between the panels, the shock can only be delivered within that area.

2.5.4 Saltwater Properties

Saltwater is a unique medium for working with electromagnetic fields. Common applications that involve the use of fields are communication networks and wireless signals. These fields usually propagate through non-conductive mediums such as air. This area of electrical engineering is formed from the basis of Maxwell's equations and the properties of alternating field propagation. This area of wireless signal transmission is one that every person interacts with every day, with little knowledge of how it works. In principle, however, electromagnetic signals are transmitted through non-conductive mediums. Saltwater has enough conductivity to be classified as a semiconductor and, as a result, it is conductive enough to act as a ground for any signal transmission. In general practice, this is the primary limitation with undersea communication. Common practices of communication simply do not work in salt water, because signals do not propagate very far.

Salt water generally describes any water that contains a high concentration of salt, roughly 2.5%. Geographically, salt water composes the oceans, seas and several larger inland bodies, in total roughly 70% of the earth's surface. The salinity levels of seawater vary with differing bodies of water. Coastal regions, affected by large freshwater supplies have lower salinities while areas with high evaporation levels, such as the Red Sea, have much higher salinity levels. Open ocean is considered substantially mixed, such that salinity levels are largely constant. Open ocean water has a salinity of between 34ppt and 37ppt (parts per thousand) (Mackenzie et al., 2011).

Freshwater is not a conductor of electricity. The conductivity of water is affected by the presence of inorganic dissolved compounds, such as salt. The higher the concentration of dissolved compounds, the higher the conductivity of the material, the easier it is for electrical current to pass through the material. Conductivity is also dependent on the temperature of the water, with warmer water being more conductive (Environmental Protection Agency, 2012). The Caribbean has an average water surface temperature of 27°C. A study showed that the exact conductivity of salt water is a function of its salinity and temperature, which is shown in Equation 1 (Ellison et al, 1998). From this, we determined, for the purposes of our experiment, trying to match conditions of the Caribbean, that the conductivity of our saltwater was $\sigma_{water} = 5.2047\,S/m$. The resistivity of a material is also the inverse of the conductivity and as a result we were able to determine the resistivity of seawater to be $\rho_{water} = 0.2952\Omega * m$.

$$\sigma(S,T) = (0.08637 + 0.03061T - 0.00041T^2) + S(0.07745 + 0.00169T + 0.00002T^2)$$

Equation 1: Saltwater Conductivity Equation

Salt water also has a high relative permittivity, a value that is a comparison of the capacitance of a capacitor using a particular dielectric with a capacitor using a vacuum dielectric. The larger the relative permittivity, the more energy can be stored by a capacitor. This value is published for many common engineering materials; however, saltwater is a far less common material for electricity. A study found the relative permittivity of saltwater at a range of frequencies, temperatures and salinity levels (Meissner and Wentz, 2004). From this study, we were able to choose the data range for 25°C with a salinity level of 35ppt. Using the data, we plotted the graph of relative permittivity vs. frequency. We then used a best-fit line to determine an equation (Figure 22) that approximates the relative permittivity at any frequency. Using this equation, we estimated the relative permittivity of warm, open ocean saltwater, at low frequencies (under 10KHz) to be 79.3.

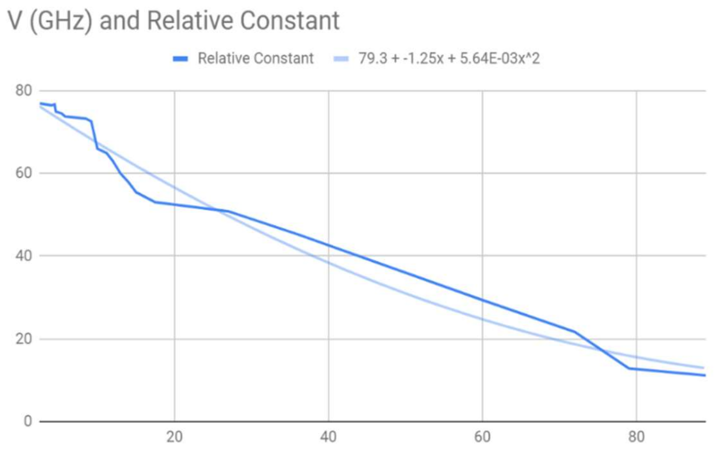


Figure 22: Graph of the Relative Permittivity of Saltwater and Frequency

2.5.5 System Circuit Models

The RSE system operates a simple circuit, composed of a battery, wiring and two steel panels. This system is functionally modeled in Figure 23. The steel plates act as both a resistive and capacitive load. The two steel plates form a simple parallel plate capacitor. The wires also offer their own small resistance values. Collectively, the circuit model of the RSE system is shown in Figure 24. For our later tests, we calculated the expected values of each of these elements. The RSE system use a 28V source and the resistances and capacitance can be calculated from our saltwater property values.

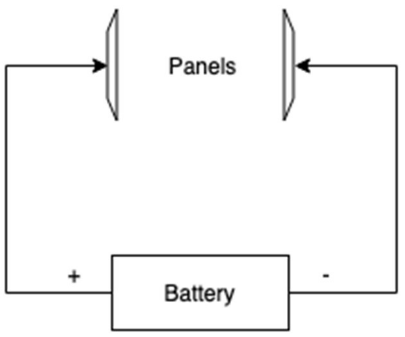


Figure 23: Electrode System Functional Circuit Model

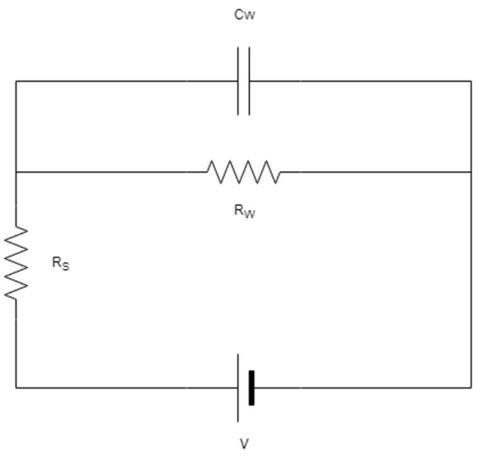


Figure 24: Electrode System Circuit Model

In order to calculate the value of the capacitor, we explored the effects of fringing on the edges of the dielectric. This effect, displayed in Figure 25, is a result of the field lines extending beyond the direct path of the parallel panels. The cross-sectional area between the plates is actually slightly larger than the panels.

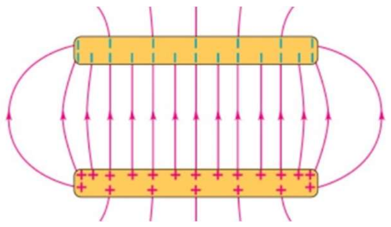


Figure 25: Capacitance Fringing Effect

In order to consider this in calculations, we needed to determine a "adjusted" cross sectional area that better describes the area between the plates. We did this using an equation, published in a study done on the fringe effect of parallel plate capacitors (Ataiiyan). This equation, shown in Equation 2, determines the adjusted cross-sectional area of a capacitor based on the capacitor's properties. We determined our adjusted cross-sectional area to be $A_{adj} = 0.1647m^2$ up from our original area calculation of $A_{cap} = 0.1016m^2$. Using this area value, we were able to determine the capacitance, shown in Equation 3.

$$A_{adj} = A * (74.5d + 0.82)$$

Equation 2: Adjusted Cross Sectional Area

$$C = \frac{\varepsilon_0 \varepsilon_r A_{adj}}{d} = 59.6pF$$

Equation 3: Capacitance Calculation

The resistance of the water between the panels can be determined with the water's resistivity value. The resistance equation is a function of cross-sectional area and distance. Using this equation in Equation 4, we calculated the resistance of the water.

$$R_{water} = \frac{\rho d}{A} = 3.78\Omega$$

Equation 4: Saltwater Resistance Calculation

The capacitor in parallel with resistor form a load with complex impedance. Using principles of complex math and circuit theory, we were able to determine the impedance load the parallel branches. The components present a 3.78Ω load. The phase shift resulting from the complex load was $4.85 * 10^{-7} degrees$. This shift indicates the negligible impact of the complex component on the much larger real resistance. Given these results, we updated our circuit model to neglect the saltwater capacitance. The consideration of wire resistance was also discounted as negligible.

2.5.6 Panel Resistance and System Current

The current distribution of the system was not as easy to calculate as the potential and field maps. The panels have their own resistivity value; that of stainless steel. The current that flows to the edge of the panels is going to encounter more resistance than the current through the center of the panel where the wire is attached. In order to model the resistance of the panels, we used a MATLAB simulation (Appendix 6.5.4) to calculate the resistance at 81 different points along a 4in by 4in square panel. Figure 26 shows the resistance values at each point.

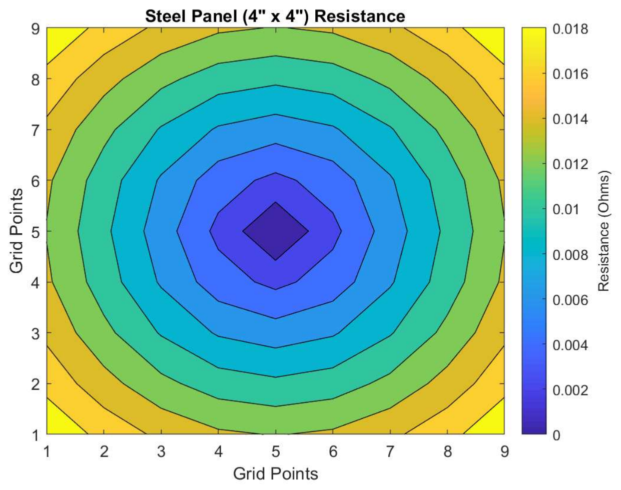


Figure 26: Panel Resistance

What we saw from this calculation, was there was a $16m\Omega$ difference between the edge of the panel and the center of the panel. In the script, the panel is broken in to 81 different areas, one for each point. The total resistance of each path is the sum of twice the steel resistance (one for each panel) and the resistance of the water path for that point. Each of these 81 paths are then summed in parallel to give us a final resistance of 3.94Ω, up from 0.16Ω from our original estimate of 3.78Ω. From each resistance, we determined the current that will flow through that path. Figure 27 shows the distribution of current across the steel panels.

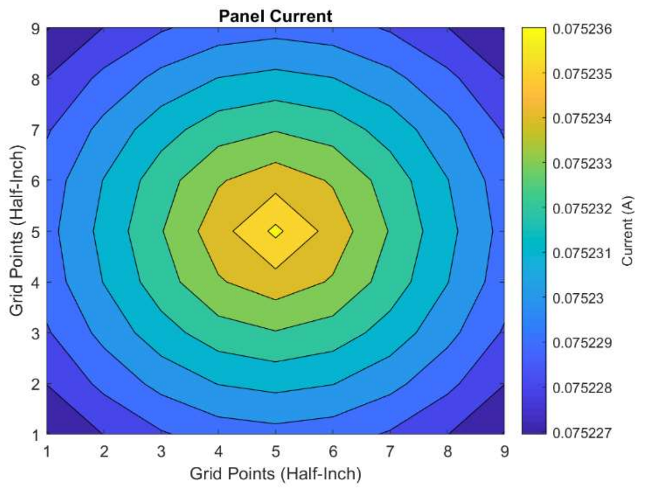


Figure 27: Current Distribution

The sum of these branches is equal to 6.09A, consistent with our original calculation of 6.34A. The decrease was to be expected as we factored panel resistance into our calculation. From this, we can also calculate the voltage change across the panel. The edge panel current and the edge panel resistance tell us there should be a 1.35mV change across the panel.

2.5.7 RSE Constraints

In addition to the goals developed in coordination with the RSE team, we were also given constraints for the system. In exploring ways to extend the field beyond the front of the platform, it was made clear that the RSE team was not interested in any mechanism that touched the fish. The shocking subsystem was to remain completely non-contact. Additionally, the system voltage had already been determined, but the RSE team was open to tweaks to the pulse configuration. Analyzing the pulse configuration, however, was outside the project goals defined during the proposal and would require additional resources not readily available to the team.

2.5.8 Adjusted System Goals

Prior to conducting our background research, we outlined three project goals for the electrode subsystem. These included documenting the shape and properties of the field, exploring methods of improving the shocking consistency of the system and exploring the possibility of projecting the field forward to shock lionfish in front of the panels. Our ability to realistically achieve the third goal became increasingly less likely as our research continued. While the panels generate an electric field in the saltwater around the system, the shock is delivered by the current flowing between the two panels. Theories of electricity and magnetism tell us that current will always flow in the shortest path with the least resistance. Because of this, the shortest path will always be directly between the two panels. In order to extend the shocking mechanism further, the conditions must be right to have the least resistive path extend forward from the system.

Several ideas were explored by the team, however, since the RSE team required that the fish not be touched, our options in this regard was limited. A non-conductive material, hypothetically, could be placed in between the two panels to change the resistance, however, such material would need to also not block the intake mechanism and would need to be large enough to block low resistive paths in every direction except forward. The team decided this was not realistically achievable and looked for alternative solutions for our third goal.

The team revised the project goals in order to take these findings into consideration. The second and third goal were merged to create a larger and more consistent shock area. This will be done by altering panel shapes and design in order to expand the shock zone while maintaining the current and voltage output of the current RSE platform. The field documentation and mapping remained unchanged. Additionally, the team looked at a method of shocking fish that differed significantly from the current design. While this mechanism was not requested by the RSE team, the team felt that its conceptual design had merit within the engineering requirements of our MQP. Its documentation by the team in Section 4.1.3 of this report is informative and can be referenced by future MQP team should they have a need to do so.

2.5.9 Stun Mechanics

There are several devices designed to stun an individual. While humans and fish have very different biological make-ups, the principles of stunning remain the same. Electrical current is passed through the target's body to disrupt electrical signals. This can be done in two ways, first, the current can simply be passed through the target, which disrupts electrical signals in the body between those two points. The body is unable to control the muscle(s) being hit and the area will generate confusing messages about what is happening. The second method is pulsing the current through the target at a specified frequency, this causes the current to mimic the signals of the target's body and will cause involuntary muscle contraction of the duration of the shock. Ideally, the involuntary contraction causes significant energy depletion and the subject is left "paralyzed" for a few seconds following the shock (Harris, 2001).

The pulsing method is the method used in law enforcement "Tasers". The devices deliver a high voltage shock of 50kV for a short period of time in order to establish a circuit through a target's clothes and force a current, even on a bad system connection with the target. Once the circuit is established, the shock itself is delivered at 1200V and has a maximum current of 1.9mA. This current is pulsed at a low enough frequency that human body resistance patterns cause the current to flow across skeletal muscles and not through deeper tissue such as the heart. The low resistance of the skeletal muscles and subsequent nervous system causes the electrical signals to extend throughout the body (Tchou and Kroll, 2007).

Using a modified version of this technique, modified for use on lionfish, it might be possible to use an extendable arm to shock a lionfish from a distance. Such a contraption would require direct contact with a lionfish, but would also extend the range of the platform and would not be concerned with the functioning of the system's electric field in the saltwater.

2.6 Intake Mechanism

2.6.1 Current RSE intake mechanism

In the RSE platform, the intake mechanism utilizes a high-power thruster to pull a stunned lionfish into the storage container. This thruster sits on the back of the robot behind the capture chamber and creates a powerful suction to intake an unharmed lionfish. In order to improve the intake system, the current lionfish team has been working with several different ideas to replace this suction system. A recent design from the group includes physical rollers that can intake the lionfish, but it has yet to be tested due to the concern they could hurt the fish. Our MQP team worked to develop a new intake mechanism that would not harm the lionfish.

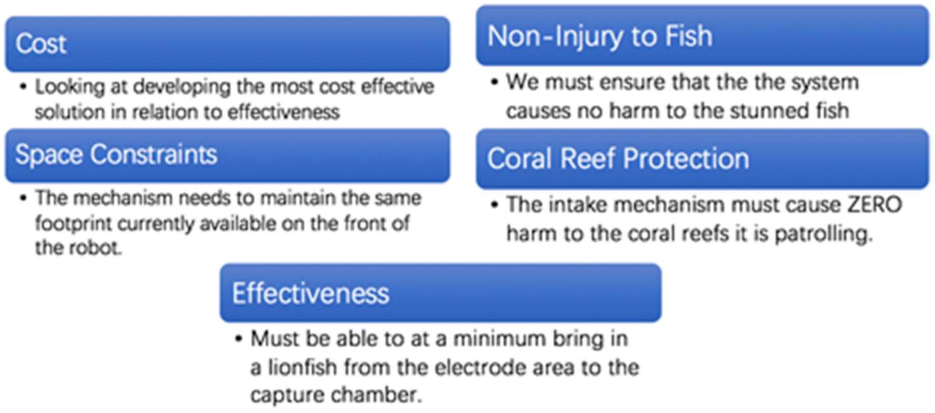

Cost
• Looking at developing the most cost effective solution in relation to effectiveness
Non-Injury to Fish
• We must ensure that the the system causes no harm to the stunned fish
Space Constraints
• The mechanism needs to maintain the same footprint currently available on the front of the robot.
Coral Reef Protection
• The intake mechanism must cause ZERO harm to the coral reefs it is patrolling.
Effectiveness
• Must be able to at a minimum bring in a lionfish from the electrode area to the capture chamber.

Figure 28: Intake Mechanism Constraints as specified by RSE

This part of the project gave us the opportunity to redesign the physical mechanism with a set of design constraints. The constraints we faced are outlined in Figure 28 and included the space available on the robot, not injuring the lionfish, not causing destruction to the coral reefs and being more effective than the Venturi pump. With these constraints in mind, we looked into other possible options for an intake device and simulate its movements. That gave us a better understanding of how it worked and what improvements we needed to make to the device.

2.6.2 Designing for a Saltwater environment

Designing in a saltwater environment requires very specialized materials in order to avoid corrosion caused by the marine environment. More specifically, we were working with a special type of corrosion called marine corrosion. Marine corrosion can be classified as many types of corrosion however most of the time its damage can be seen through a material’s contact with salt water. The seawater solution, when exposed to oxygen, becomes very acid and anodic and starts corroding the material through tiny cracks and crevices leading to overall corrosion. To compound the issue, different organisms can affect the acidity level of the seawater and the sand and silt can erode materials (304 stainless steel in seawater, 2017). Materials such as polyurethane and polycarbonate are commonly used in marine applications due to the fact they are non-reactive and non-corrosive, which leads them to be the better material for underwater use (Polyurethane for marine applications). However, many times metal is required for a project. So, the marine standard has become stainless steel which has a very low corrosion-erosion rate (Wood, 2006). Unfortunately, non-corrosive materials tend to have slightly higher price tags making the entire project more expensive.

2.6.3 Design Considerations

The lionfish intake mechanism is essential to the functionality of the entire device. A simple and efficient capture and contain design needs to transport the stunned lionfish from in between the electrode panels into the body of the device. This design needs to not only meet the initial design requirements as seen in Figure 28 but also meet the more defined goal of making drastic changes and improvements. The criteria that deem this success are ease of use, repeatability, speed, and efficiency. From these requirements, we designed several ideas with a myriad of different approaches to intake lionfish which varied between direct mechanical collection via conveyor or linear slides along with the use of suction and jet.

2.6.3.1 Linear Slides

One of the first ideas that came to mind for capturing lionfish was with the use of linear slides. The linear slides would have a one-way gate that allows the lionfish to enter the area between the two electrode panel’s arms, by using the force of propelling the linear slides forward to open the one-way trap. Once the fish has been captured in the trap, the linear slides will actuate so that it forces and contains the lionfish in the container. These linear slides would be mounted to the sides of the containment container and offer a minimal interaction capture mechanism that would not damage the fish. This design can be seen in the drawing below in Figure 29.

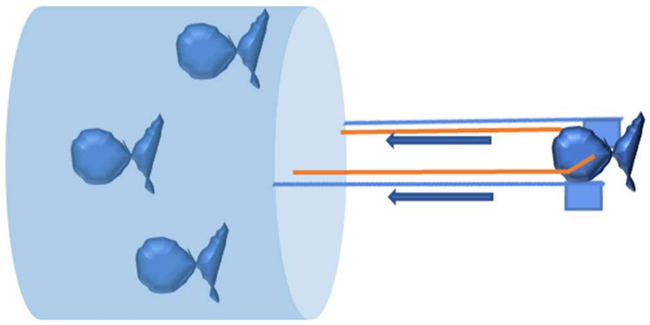


Figure 29: Initial Linear Slides Design for Capturing and Containing Lionfish

2.6.3.2 Conveyor Belt

This design was based off a similar concept posed by a volunteer with RSE. The RSE engineer designed and 3D printed rollers that would intake the lionfish, however, due to the fact it looked to be a possible danger to the fish it was not tested on the robot. Therefore, our design considered the ultimate safety of the fish. In our design, we had two conveyor belts rotating in opposite directions with soft finger-like pieces attached to the belt to softly corral the fish in. These finger-like pieces, which were inspired by sea-anemones, would prevent any escaping from the containment device. A CAD model without the soft pieces attached to the belt can be seen in Figure 30.

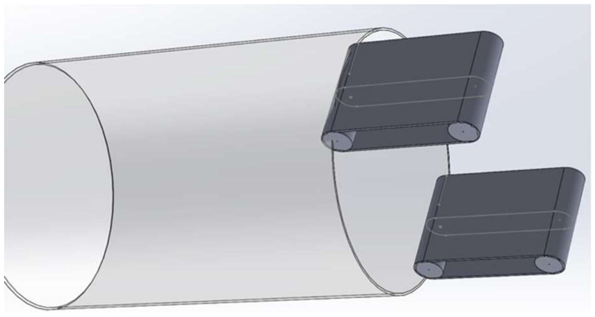


Figure 30: Initial Conveyor Belt Design for Capturing and Containing Lionfish

2.6.3.3 Suction

As inspiration, we also researched many unique methods of intake for fish which included a device called a "salmon cannon" (Wooshh Innovations, 2018). A salmon cannon is an innovative solution to man-made dams that prevent the migration of salmon due to their high walls. This system uses a pressure differential to create a suction to transport salmon from small collection ponds to their desired destination. These fish can be transported very long distances of up to 20 MPH with no effect on the fish. An incentive to this solution is its indirect method of capturing fish that is not totally different to what exists currently on the system. In our case, we will need to contain the fish in a holding chamber with a one-way opening. After exploring this idea, research was conducted to determine a way to utilize changes in pressure to create a velocity and suction leading us to the Venturi effect.

The Venturi effect is a phenomenon created by a Venturi meter that was discovered in the early 1800s by Giovanni Battista Venturi (Venturi, 1799). It is created by the "relatively streamlined contraction (which eliminated separation ahead of the throat) and very gradual expansion downstream of the throat" (Gerhart, 2016) of the pipes as seen in 1 and 2 respectively of Figure 31 below. The shape of this device allows for the reduction of head losses due to friction and efficient mixing in well-designed Venturi meters. Simply, this means that the tubes contraction causes the flow rate to change due to a change in pressure from 1 to 2, which can be seen in Figure 31. This flow rate out Q, exiting velocity v2, and exiting pressure is affected by the initial tube's area A1, velocity v1, pressure p1, environmental pressure ρ, and the constricted tube's area A2.

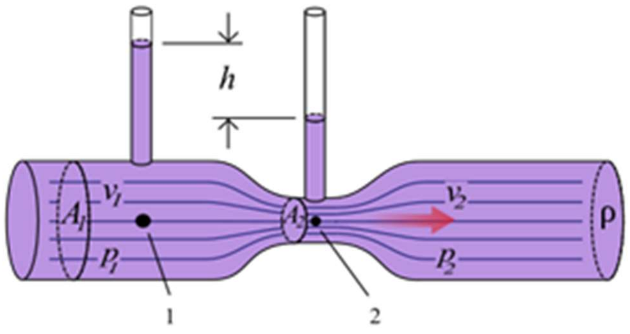


Figure 31: Picture depicting the flow of fluids through a Venturi meter

This effect is highly driven by Bernoulli's equation, as seen in Equation 5, which in its simplicity says that all the energy in must equal the energy out. Therefore, the pressure energy, kinetic energy, and potential energy from one point must equate to the other point.

$$P_1 + \frac{1}{2}\rho v_1^2 + \rho g h_1 = P_2 + \frac{1}{2}\rho v_2^2 + \rho g h_2$$

Equation 5: Bernoulli's Equation (Conservation of Energy)

This "Bernoulli effect" is the actual phenomena that allow for an increase in velocity in the pipe through the reduction of pressure in the tube by reducing the area in which the fluid can flow. One can see the relationship pressure and rate of flow "Q" through the equations in Figure 32.

$$Q = v_1 A_1 = v_2 A_2 \qquad p_1 - p_2 = \frac{\rho}{2}(v_2^2 - v_1^2)$$

$$Q = A_1 \sqrt{\frac{2}{\rho} * \frac{(p1 - p2)}{(\frac{A_1}{A_2}^2) - 1}} = A_2 \sqrt{\frac{2}{\rho} * \frac{(p1 - p2)}{1 - (\frac{A_2}{A_1}^2)}}$$

Figure 32: Venturi Equations Derived from Bernoulli's Equations

We designed what this attachment may look like in Figure 33 and it seems that this will be a very small attachment that can replace the current thruster. It will need a pump to create the pressure differential in the intake. That differential pressure would create a powerful suctioning force that would intake the lionfish into the holding chamber and with its configuration we would be able to not lose suction with the increase of lionfish.

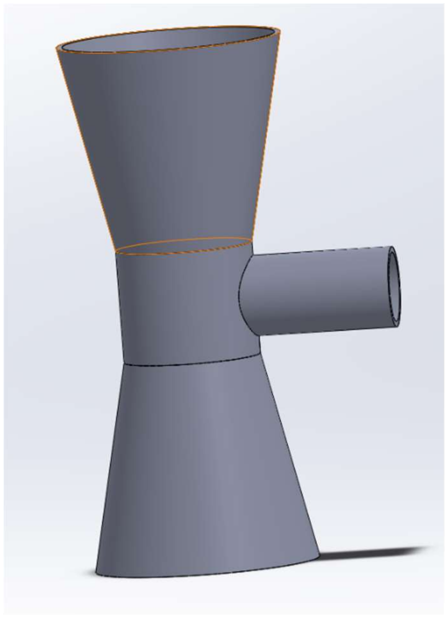


Figure 33: Initial Venturi Design for Capturing and Containing Lionfish

3. Methodology

3.1 Electrode System

3.1.1 Panels

The panels currently used by the RSE team are constructed with 22-gauge stainless steel (0.025in thickness). The panels are each 4in tall by 4in wide, with a combined surface area of 16in^2 or 0.0103m^2. Our previous calculations show that this panel size results in a water resistance of $R_{water} = 3.78\Omega$. Given the 28V source and a water resistance (neglecting wire resistance), we calculated the expected current of the system. Shown in Equation 6, this is the amount of current, that when passed through the 0.0103m^2 panels results in a successful lionfish shock.

$$I = \frac{V}{R} = 6.565A$$

Equation 6: System Current Calculation

However, this value is slightly misleading. The panels cover an area much larger than a wire and the position of a lionfish within this area will have an impact on the effectiveness of the shock. Instead of looking to maintain a consistent current value across our several panel shapes, the team looked to keep the current density across the varying panels consistent. This is to say that the current per square meter was held constant. The calculation for current density is shown in Equation 7. The team designed the different panels to maintain a 0.0103m^2 conductive surface area. As a result, the area, current density, and current all remained relatively constant.

$$J = \frac{I}{A} = 635.98\, A/m^2$$

Equation 7: Current Density Calculation

The team designed four panel shapes to explore the effect of panel shape on the field and identify the panels that offered the largest overall current effective area. The team's working hypothesis was that a larger effective shock area would require system operators to be less precise in their alignment of the panels relative to a fish, in order to deliver a successful shock. The fifth set of panels were constructed at a 50% scale model in order to view a larger area of field surrounding the panels. The panels shapes, shown in Figure 34, were constructed out of 16-gauge stainless steel, similar to the material used by the RSE team.

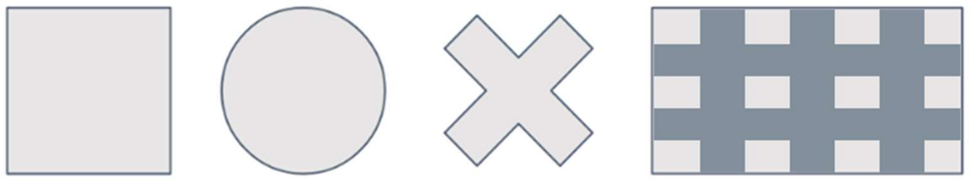
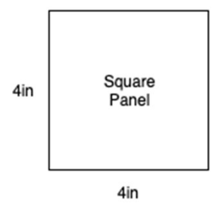
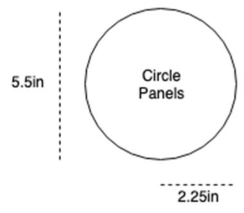
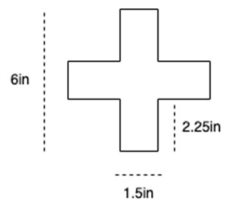
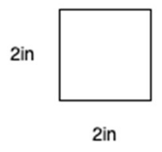
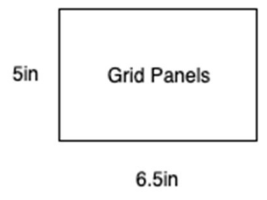


Figure 34: Proposed Panel Shapes

The panels were sized to closely match the size of the original panels. The square panels (Table 1) were cut identical to the existing RSE panels. The circle panels (Table 1), were cut with a radius of $r = 0.0572m$ or 2.25in. The larger grid shape panels (Table 1) were cut to have a width of $w = 0.127m$ and a length of $l = 0.1651m$ or 5in wide by 6.5in in length. The small panels (Table 1) were cut to have four equal sides of $w = 0.0508m$ or 2in. The cross-shape panels (Table 1) were cut such that each "arm" of the cross had a width of $w = 0.0572m$ or 2.24in and a length of $l = 0.0381m$ or 1.50in. The resulting surface area of each panel is shown in Table 1.

Panel Shape	**Area (in^2)**	**Image**
Square	16.00	
Circle	15.90	
Cross	15.75	
Small	4.00	
Grid	32.50[*]	

[*] Non-conductive material adjusts for a 16in^2 conductive surface area

Table 1: Panel Shape Areas and Designs for the Panels

The panels were cut using a machine sheer in the WPI materials shop. The panels that had square corners were easily cut, however, the cross shaped panel required a level of precision that was achieved using a hacksaw. The circle panels also proved difficult to cut accurately.

The final shape was achieved using a sheer press and a hacksaw. Both the cross panels and the circle panels were filed down in pairs in order to obtain a uniform shape. The circle and cross panels were both covered on the back side with non-conductive material in order to compare the field with those uncovered. The grid panel was also covered on the front with non-conductive material as seen in Figure 35. This shape resulted in a conductive surface equal to the desired amount.

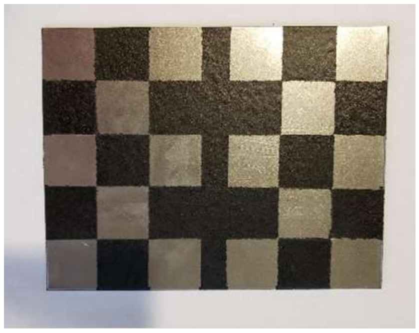


Figure 35: Grid Panels

On the back of each panel, we attached a small metal component. This metal attachment was secured to the panels using *E6000 Industrial Strength* Adhesive. On top of this metal component, we secured a K'nex part, which was used for attaching the panel to our testing setup. The wire for the testing was then wrapped around the metal screw and secured between two nuts. The completed panels can be seen in Figure 36. The entire attachment can be seen in Figure 37. Each one was then used in testing.

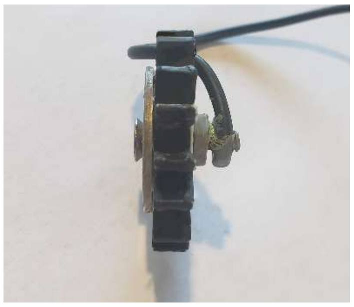


Figure 36: Panel Attachment

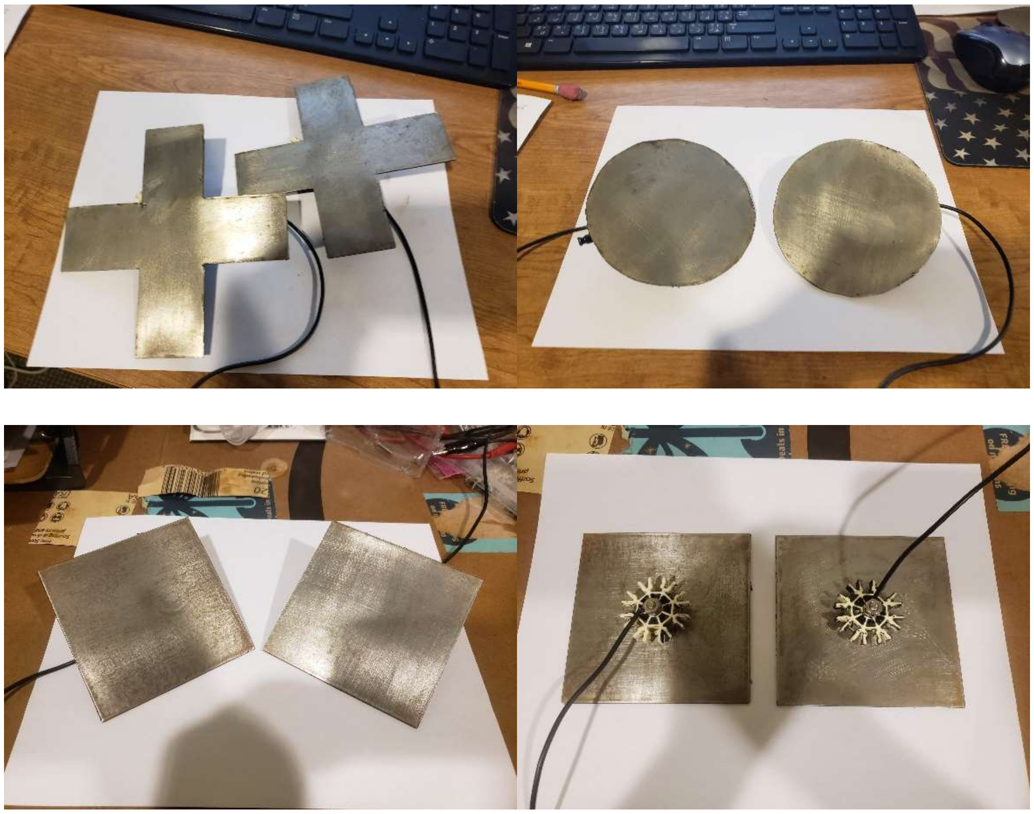


Figure 37: Completed Panels

3.1.2 Saltwater

In order to conduct out testing, we need a saltwater basin where we could control the salinity levels. The desired salinity level was 35ppt, with a tolerance level of 1ppt. The team determined that we needed to use approximately 10 gallons of saltwater. In order to achieve the desired concentration, we needed to add 35 grams of salt for every liter of water. For simplification of measurement, the team added 136.5 grams of salt to each gallon of water. Each gallon of water was brought to a boil on a household stove and then a half cup (136.5 grams) of salt was added slowly until it was completely dissolved in the medium. This resulted in 1365 grams of salt mixed with $37.85l$ (10 gallons) of water. At these measurements, our saltwater solution was 36ppt. In order to decrease this, we added one additional liter of freshwater, resulting in $\approx 38.85l$of water mixed with 1365 grams of salt, or 35.1ppt. During the course of the experiment, the water was maintained at a room temperature of 22°C.

3.1.3 Electrical Configuration

The testing mechanisms were powered by two 12V car batteries. The lead-acid batteries were the best solution given the amperage the simulations predicted. The batteries were connected in series and initially measured a source voltage of 24.84V. By the end of the testing, the batteries had decreased slightly to 24.8V. The system was wired using 18-gauge copper wire. The wire was chosen for its availability and the fact that it had a current rating of 16 amps, well within our predicted current requirements. The positive wire terminal was then run through a household wall switch. This allowed the electrical system to be engaged and disengaged quickly. The ground panel was connected directly to the ground terminal of the battery component. A block diagram of the system can be seen in Figure 38. This simple circuit allowed for the system to be "pulsed" long enough to take a voltage measurement and then be turned off again to conserve the charge of the batteries. Using a digital multi-meter (DMM) we were able to measure the voltage and the current of our circuit.

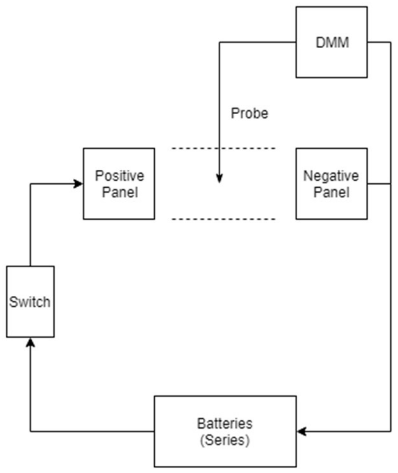


Figure 38: Electrode System Electrical Block Diagram

3.1.4 Basin Setup

One of the most notable challenges the team faced during the electrical testing, was the way to secure the panels in the saltwater basin. Several ideas were put forward, ranging from fixing the panels to a metal rod through the side of the container to hanging them from above with string. Each option presented its own drawbacks. Going through the side of the basin present the issue of leaks and dealing with underwater seals. The string from above presented the issue of being too flexible for accurate positioning. Ultimately, the team settled on a design that utilized K'nex parts. These simple plastic building blocks proved rigid enough to maintain the panel's position, however, adaptable enough for the needs of the project. The holders were run through the side of the plastic container, above the water line. They were then fixed in place using K'nex pieces on the front and back of the plastic joint. A picture of the empty holders can be seen in Figure 39. Holes were drilled at specific locations at the top of the container in order to allow for the holders to be fixed 8in apart and 4in apart. Each holder rested on the bottom of the plastic basin and was braced against the side of container at the bottom. This is shown in Figure 40. This prevented the panels from sagging the holders. The holders were positioned vertically to hold the panels in the middle of the saltwater depth. This allowed for roughly 1in of saltwater underneath the panels and 1.5in of saltwater above the panels.

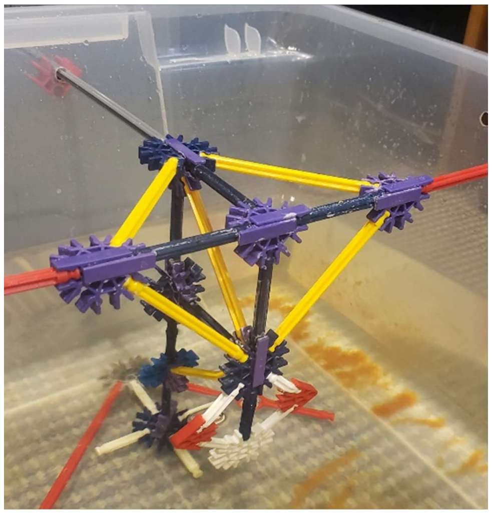


Figure 39: Saltwater Basin Empty K'nex Panel Holders

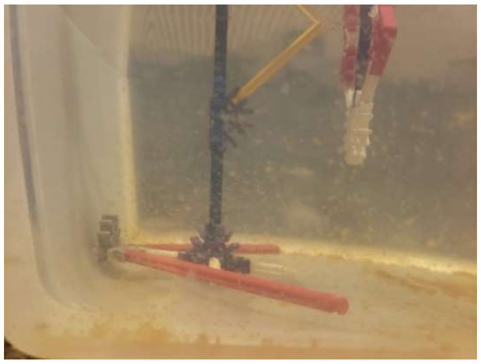


Figure 40: Saltwater Basin Panel Holder Stands

The measurements were taken using a wire probe affixed to a custom Cartesian tool. Seen in Figure 41, the bridge mechanism sat along the top of the plastic basin. The tool slid freely along the top of the container and contained an arm that slid along the width of the container. Finally, the wire probe was attached to a rigid pole that slid vertically up and down. Each axis allowed movement freely enough to easily reposition the tool, however, held in place with enough force that it did not move while the measurement was being taken. The range of the tools motion extended the entire width, length and depth of the container. Along each axis, positions were marched out in Sharpie marker. The top of the container contained marks every inch as seen in

Figure 42. The other two axis had points marked out along the fixed axis rod. This allowed the repositioning of the container to be done easily without measuring the location each time.

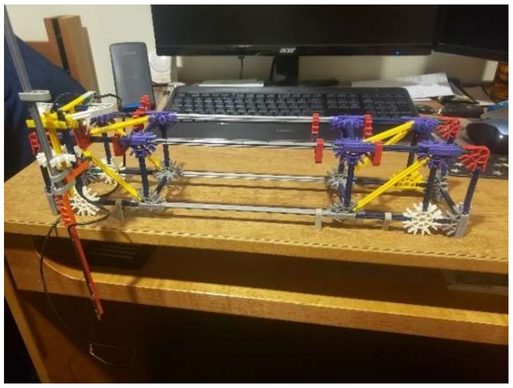


Figure 41: Saltwater Basin Measurement Bridge Tool

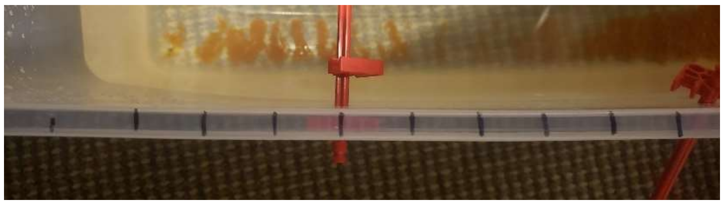


Figure 42: Saltwater Basin Distance Markings

The wire used for the probe was simple copper wire, the same type used for connecting the different components of the system. The team was concerned of water leakage underneath the rubber wire insulation. Should water leak under the insulation, it could have unforeseen effects on the measurement. In order to address this risk, the team capped the end of the wire, save the 5mm left bare, in silicone. This proved to help prevent measurement issues due to water seepage. The silicon encased wire ending can be seen in Figure 43.

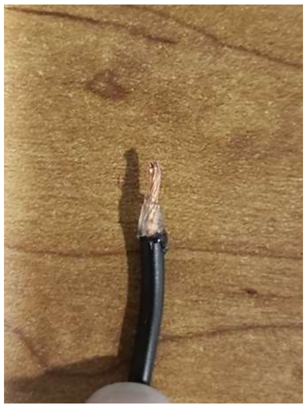


Figure 43: Saltwater Basin Measurement Probe Silicon Encased Wire

3.1.5 Testing Method

The panels were each tested in an organized manner in order to maintain consistency in our results. The panels were connected to the holders and then the wires were each attached to their respective locations. Once the system was connected, the measurement tool was set on top of the basin. The tool was moved to the appropriate location on all three axes and the measurement lead was connected to the positive end of the DMM. For our voltage measurements, we used a *FLUKE 75 Series II Multimeter*. Once the system was ready, a short pulse, triggered by the changing state of the switch, was initiated and a voltage measurement was recorded in a spreadsheet, with corresponding X, Y and Z location. The axis corresponding to X, Y and Z can be seen in Figure 44.

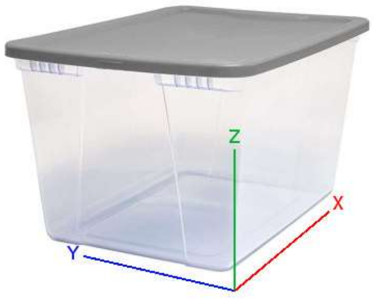


Figure 44: Axis Relative to Container used as Saltwater Basin

Measurements were taken at 84 points for each set of panels. There were 3 Z-locations, 4 Y-locations and 7 X-locations. Z-measurements were taken at 0in, 3in and 6in. Y-measurements were taken at 0in, 4.5in, 8.5in and 13in. X-measurements were taken at 0in, 2in, 4in, 8in, 12in, 14in and 16in. The grid location of measurements relative to the panels, can be seen in Figures 44, 45 and 46.

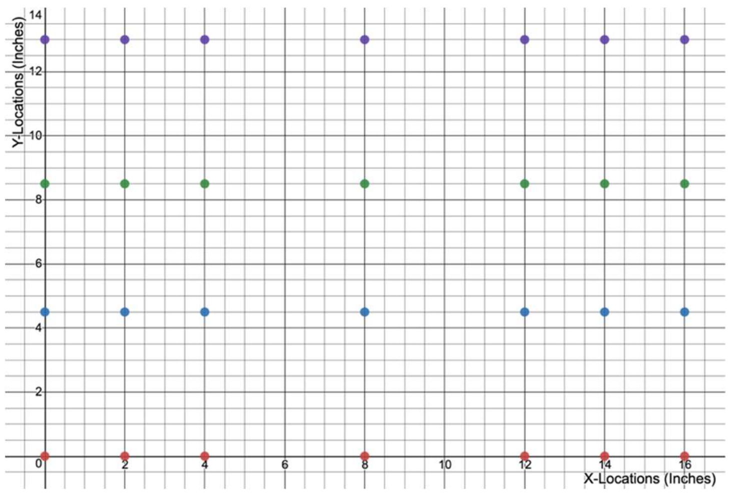


Figure 45: Saltwater Basin X-Y Locations with Panel Location (Yellow)

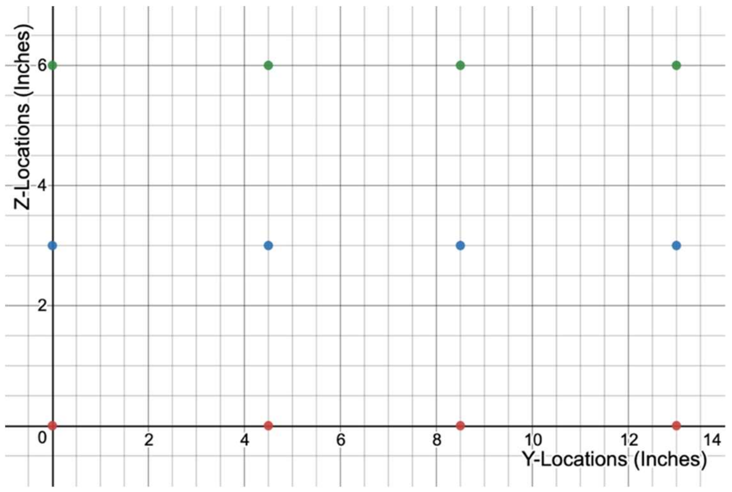


Figure 46: Saltwater Basin Y-Z Locations with Panel Location (Yellow)

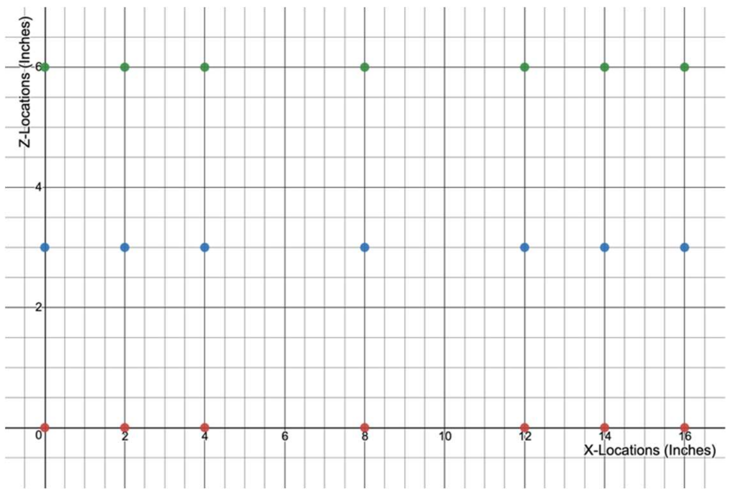


Figure 47: Saltwater Basin X-Z Locations with Panel Locations (Yellow)

3.1.6 Testing Observations

Throughout the several rounds of testing, several things were consistently observed. First, saltwater proved to be an extremely harsh medium of operation. Panels consistently showed extensive signs of wear. The positively charged panel incurred discoloration of red hues and the negative panel was worn to a tarnished appearance. The panels oxidized within minutes of removal from the salt water. Immediate drying of the panels helped slow but did not prevent this oxidation process. The copper wire also showed significant signs of wear and oxidation where it was exposed to the saltwater. Similarly, the glue used in the construction of the panels held up only in the nicest of situations. While statically attached to the holders, they maintained their position and electrical connections, however, as soon as the panels were removed from the saltwater, the slightest pressure on the glued attachment resulted in the glue adhesive failing. The glue was rated for use in water, however, saltwater proved to be more than it could handle. When retesting occurred, the panels needed to be reconstructed.

Another significant observation during our testing was the effect the panels had on the saltwater itself. After measuring the first layer of our first set of panels, the water had begun to exhibit a light brown hue. This continued to worsen as we continued to measure the voltage points. It increasingly got a “cloudier” look, and ultimately turned a dark rust color. The solution sat overnight between our first and second day of testing. We observed the following morning that a thick layer of particulates had settled to the bottom of the container and the saltwater solution was once again observably light brown, but clear enough for visible light to travel through it clearly.

Testing needed to be paused in order for us to adjust our panel construction. Due to the existing delay, we opted to drain the murky saltwater and create another 10 gallons of water, using the same process as the first time. The effect on the water was identical to the first time and testing continued. When current measurements were taken for the third time, a third batch of saltwater was used. The similar process occurred again. The most logical explanation for this, is a corrosive reaction between the panels and the saltwater. The rust colored water is a result of particulates being expelled from the plates during their time submerged in the corrosive saltwater. This addition of particles in the water would also help explain discrepancies of water resistance over time, as the particulates would decrease the water’s resistance property.

3.1.7 Data Collection

Voltage measurements were initially consistent with simulations, however, by the second layer of testing on the first set of panels, the results were beginning to look odd. By the third and final layer of the first set of panels, there was completely inaccurate readings being recorded. Further inspection indicated that the panels were separating from their connectors and the wires were only intermittently touching the panels. In order to fix this problem, all the panels were reconstructed in a more durable manner, the processes described in Section 3.1.1. The second version of the panels were significantly more durable and held up for the duration of testing. The grid panels, however, proved too heavy for the connection method and fell apart half way through voltage measurements. Given the team's timeline and otherwise good results, we opted to forego the testing on the final set of panels.

Current measurements were taken using both the *FLUKE 75 Series II Multimeter* and the *Extech EX410: 8 Function Professional Multimeter*. The saltwater was also replaced between voltage measurements and current measurement. The original round of current measurements indicated between 9.7A and 9.8A for all panels. These values seemed inconsistent with the simulations, as the smaller panels should have resulted in higher resistance and lower current than the large panels. The tests were run again in a clean batch of saltwater and the current was measured to be approximately 12A. These values were initially accepted as correct, until the same issue was encountered toward the end of testing. The small panels again showed the same current as the larger panels. The team ran the tests a third time using the second multimeter and results, while outside the simulated expectations, were consistent with the fact that the smaller panels would encounter more resistance due to the small area.

The final round of current measurements was also taken by attaching alligator clips directly to the panel, instead of using the copper wire. This was done due to the corrosion that had built up on the copper wire from the saltwater. We discovered the use of the copper wire and attachment component made a significant impact on the amperage. When the 4ft of copper wire and panel connection attachment was used, it effectively doubled the resistance of our system. This finding, while not accounted for during our simulations is one that makes sense. The attachment was affected by the harsh effects of the saltwater and the slightest gap between the connector and the panel would cause a resistive value to affect the circuit. Given the already low resistance offered by the saltwater, the connectors could have substantially changed the measured values with only 1-2Ω of resistance. This was demonstrated in the readings of the large square panels where we measured a current of 7.8A when connected through the wires and 15.8A when connected with only the alligator clips. The remaining large panels, through the alligator clips, showed currents of 15.85A for the cross panels and 14.6A for the circle panels. Ideally, these values were all be within a few milliamps of each other. The construction of the circle panels, however, was slightly less accurate than the three, due to the nature of the tools available to the team. After the circle panels were filed to shape, the surface area proved to be slightly smaller than anticipated and the smaller current value measured is consistent with that fact.

The current readings are discussed in Section 3.1.10 and the results of the small panel tests are discussed in Section 3.1.11. Tables 2 contains the collected measurements at each location for the large square panels. Only one panel is shown for reference, however, the remaining data points can be found in Appendix A.4.

X	Y	Z		Volts
0	0	0		16.74
0	4.5	0		17.35
0	8.5	0		17.42
0	13	0		16.7
2	0	0		16.33
2	4.5	0		17.5
2	8.5	0		17.62
2	13	0		16.32
4	0	0		15.21
4	4.5	0		16.74
4	8.5	0		17.12
4	13	0		15.07
8	0	0		11.96
8	4.5	0		12.03
8	8.5	0		12.02
8	13	0		11.89
12	0	0		8.83
12	4.5	0		6.5
12	8.5	0		7.16
12	13	0		8.64
14	0	0		7.43
14	4.5	0		5.96
14	8.5	0		5.98
14	13	0		7.27
16	0	0		7
16	4.5	0		6.54
16	8.5	0		6.22
16	13	0		6.87

0	0	3		16.81
0	4.5	3		17.38
0	8.5	3		17.47
0	13	3		16.81
2	0	3		16.49
2	4.5	3		17.74
2	8.5	3		17.87
2	13	3		16.47
4	0	3		15.37
4	4.5	3		19.42
4	8.5	3		19.9
4	13	3		15.47
8	0	3		12.5
8	4.5	3		12.64
8	8.5	3		12.74
8	13	3		12.49
12	0	3		8.66
12	4.5	3		3.51
12	8.5	3		3.85
12	13	3		8.44
14	0	3		7.41
14	4.5	3		6.04
14	8.5	3		5.85
14	13	3		7.17
16	0	3		7.02
16	4.5	3		6.38
16	8.5	3		6.27
16	13	3		6.85
0	0	6		16.84

0	4.5	6		17.31
0	8.5	6		17.4
0	13	6		16.72
2	0	6		16.46
2	4.5	6		17.31
2	8.5	6		17.52
2	13	6		16.38
4	0	6		15.35
4	4.5	6		16.94
4	8.5	6		17.49
4	13	6		15.36
8	0	6		12.61
8	4.5	6		12.34
8	8.5	6		12.7
8	13	6		12.46
12	0	6		8.62
12	4.5	6		6.57
12	8.5	6		6.8
12	13	6		8.59
14	0	6		7.37
14	4.5	6		6.15
14	8.5	6		6.35
14	13	6		7.4
16	0	6		7.02
16	4.5	6		6.34
16	8.5	6		6.39
16	13	6		6.97

Table 2: Square Panel Voltage Measurements

3.1.8 Electrical Potential Mapping

Using the collected voltage data from each panel, we developed MATLAB scripts to plot the electric potential contours. The script (Appendix A.5), maps each data point, interpolates an additional 45 points and maps the voltages across the plane. Each set of panels has 27 potential graphs. A complete result set is found in Appendix A.1, however, the summary of results is contained within this section.

The plots represent three unique views of the system. Figure 48 shows the view of the potential map from a top down perspective. Figure 49 is a cross sectional slice of potentials in line with the panels. Figure 50 is a side view perspective of the electric potential at each point. Each of these figures are the standard large square panels.

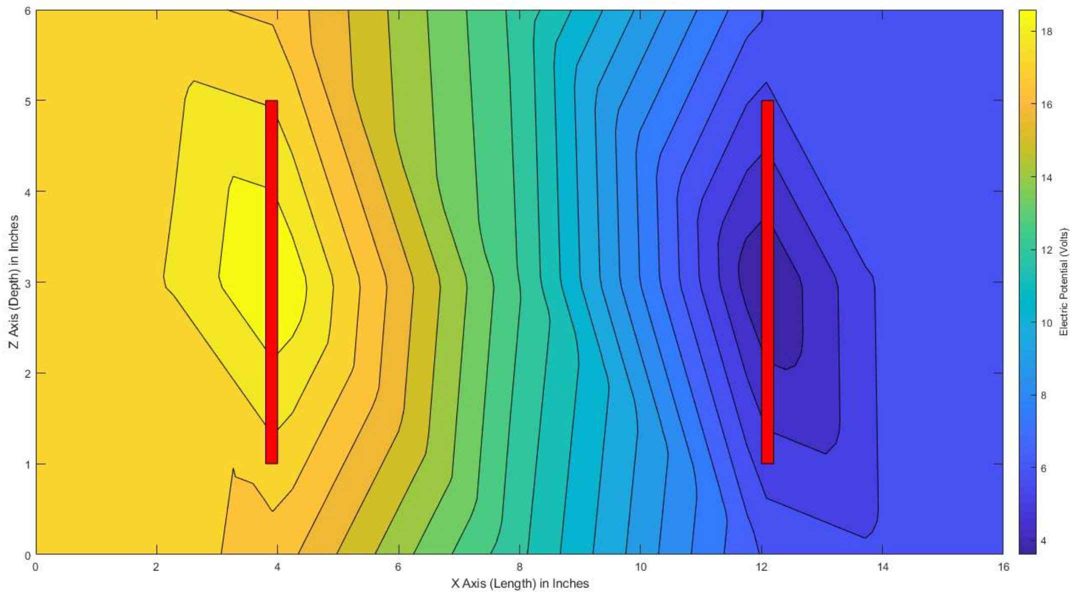


Figure 48: Measured X-Z Potential Map

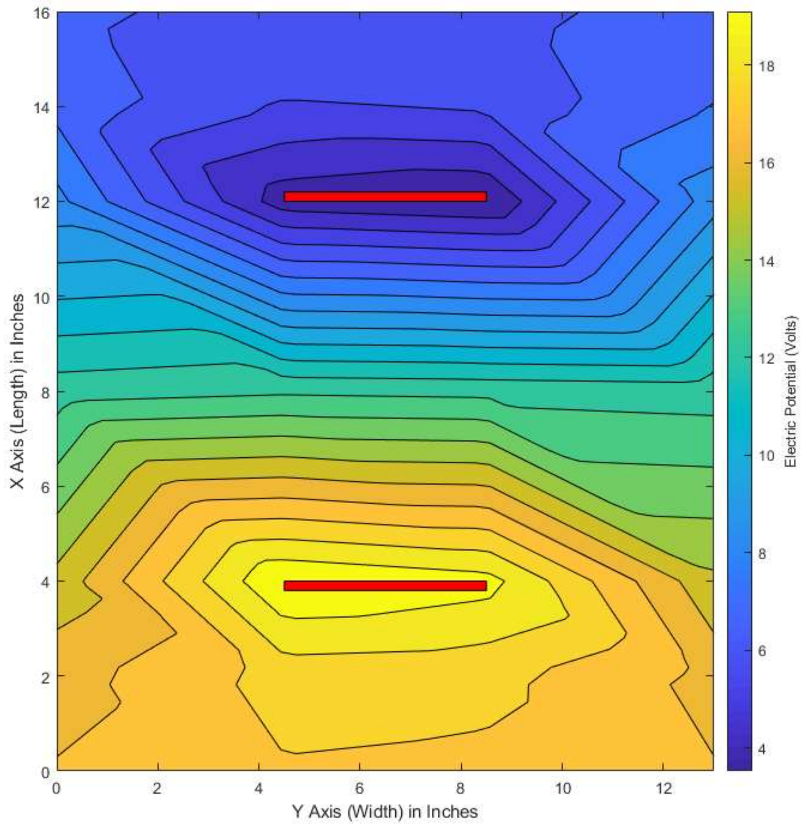


Figure 49: Measured X-Y Potential Map

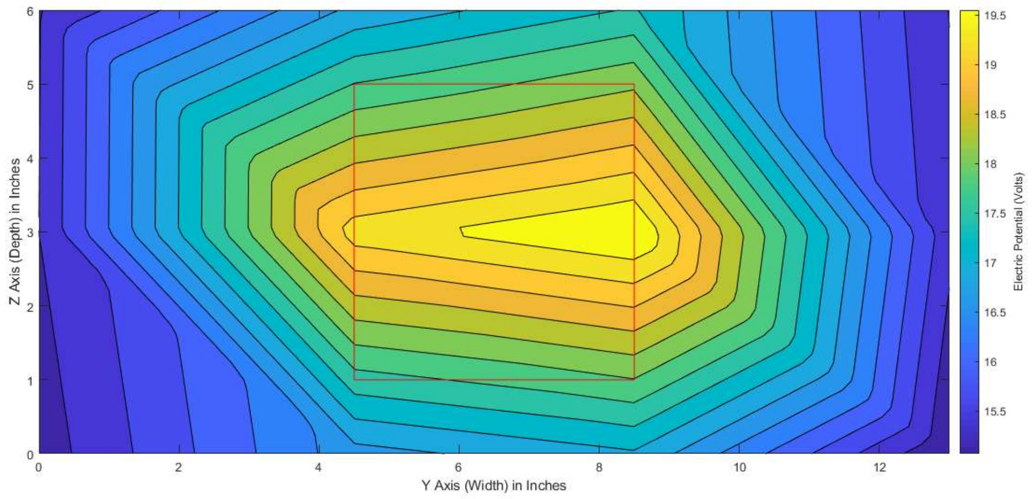


Figure 50: Measured Y-Z Potential Map

The team found the X-Y (top-down) plots, there were taken at the middle depth (3in) were the most helpful of the figures generated. This is because the measurement at this depth was taken in line with the panel, while the other depths were below and above the panel. The in-line measurement offers the reader a good sense of the field shape. The X-Z (vertical) plot in Figure 48 is a good example of the overall resolution that was lost due to measurement points chosen by the team. The figure shows a high and low voltage at the center of the panel. The panel itself, however, was charged to the same voltage and as a result, the entire area surrounding the panel should indicate as such. The extremity of the field was measured with high accuracy; however, middle segments experienced some data loss, since our focus was directed toward the edges.

3.1.9 Electric Field Shape

Deriving electric field strength from electric potential is a trivial task. The value of the field is equal to negative the change in potential over a given distance. This is expressed in Equation 8. The direction of the field vector is orthogonal to equipotential lines, or contour lines. This means that from our potential maps, we can easily calculate the electric field at every point and construct a plot of the field.

$$\underline{E} = -\left(\frac{dV_x}{dx}\hat{x} + \frac{dV_y}{dy}\hat{y} + \frac{dV_z}{dz}\hat{z}\right)$$

Equation 8: Electric Field Vector Equation

The field shape is consistent with the dipole model we hypothesized during our research. In the event of an alternating field, such as the one implemented on the RSE platform, this shape represents the steady-state field. The field is centered around the horizontal bars, instead of single points. The field extends outward from the positive panel and converges on the ground panel. The field is uniform between the two panels and shows a consistent loss of potential, as expected.

Outward from this channel, the field lessens. Figure 51 through 52 show the mid-level, top down field for each set of large panels.

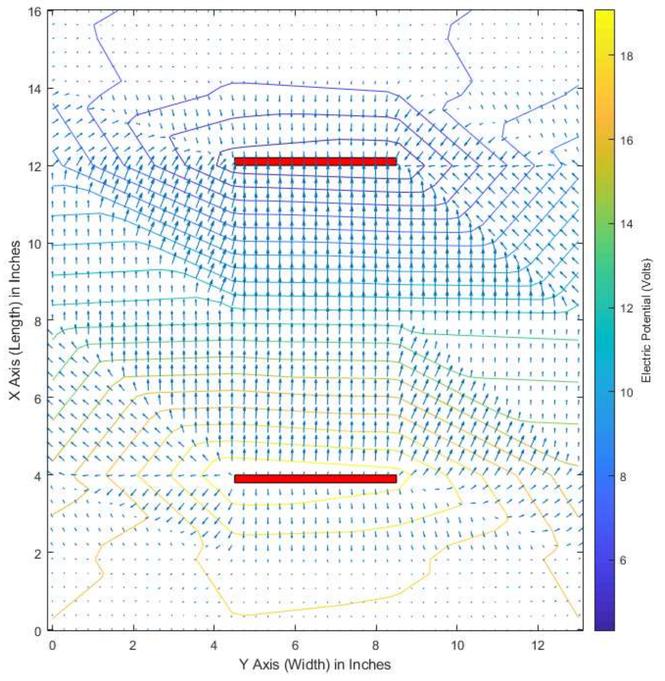


Figure 51: Square Panel – Mid-Level – Top Down Field

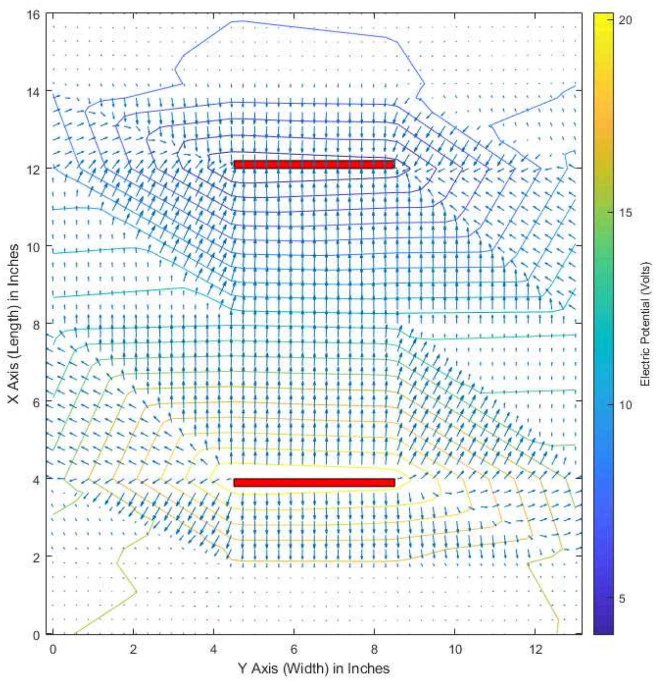


Figure 52: Circle Panel – Mid-Level – Top Down Field

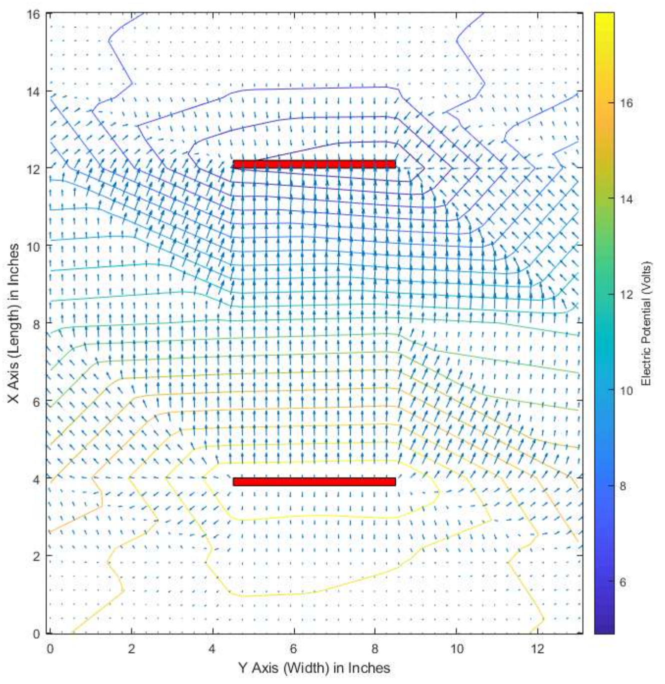


Figure 53: Cross Panel – Mid-Level – Top Down Field

3.1.10 System Current

As current flows between the two panels, there is a consistent voltage loss. The voltage drop across any given area is equal to the current flowing times the resistance between the points. This equation is shown in Equation 9.

$$\Delta V = IR$$

Equation 9: Voltage Law

Our simulation shows widely uniform current distribution. Showing that the panel had consistent voltage across its face, confirms our hypothesis that the current across the panel is largely equal. This hypothesis is also reinforced by the fact that a uniform field exists between the two panels as shown in Figure 54.

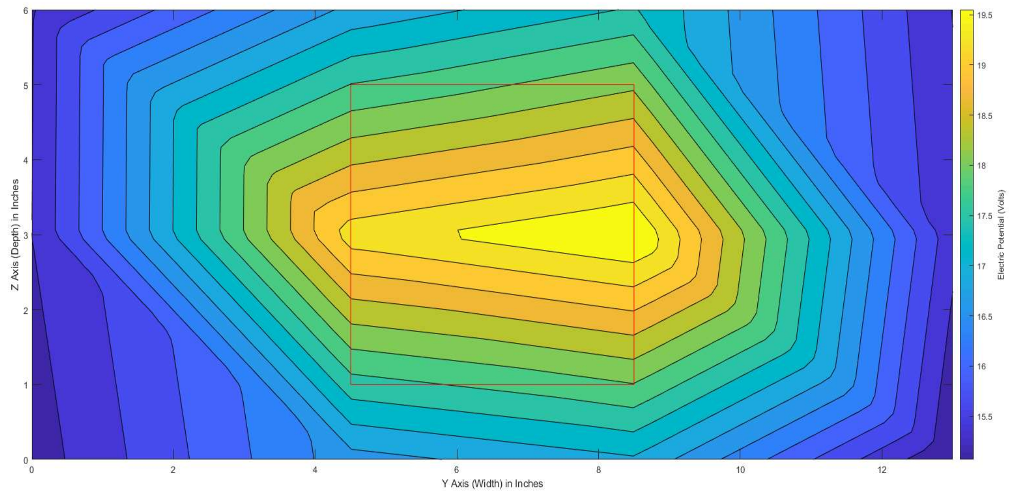


Figure 54: Square Anode Cross-Cut Voltage Distribution

The voltage map of the square panel points closest to the anode is shown in Figure 54. The six data points of concern are the two taken at the bottom of the graph (1in below the panel) at Y=4.5in and Y=8.5in; the two taken at the middle of the graph at the same Y-points; and the two points taken at the top of the graph (1 in above the panel). Since we do not have measurements at Z=2 and Z=5, the equipotential lines are determined from a gradient. Averaging these six points out gives us a largely uniform voltage distribution. The normalized voltage map is shown in Figure 55.

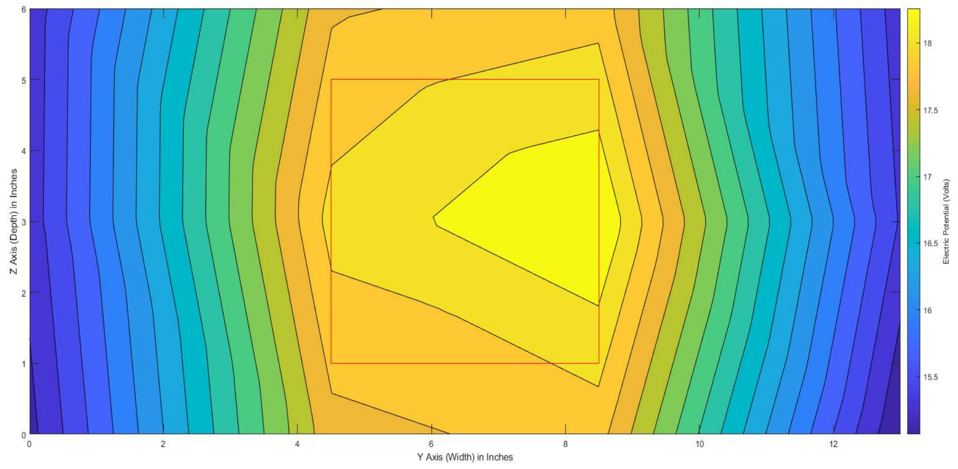


Figure 55: Square Anode Cross-Cut Voltage Normalized Distribution

The red line in Figure 55 shows the location of the panel in the distribution. While there are three contour lines across the red rectangle, the total voltage change from the lowest to highest voltage of the panel is $\Delta V = 0.54V$. This value would likely be much closer to 0V if we had taken more points along the edges of the panels.

Consistent with our simulation, the voltage distribution across the plate appears to be largely uniform. From this, we can determine that the current across the plate was also distributed largely uniformly, which was expected. Working on this supported the hypothesis that the determining factor in an effective shock is the amount of current that is passed through the fish, not where the fish is located over the panel. This is also consistent with our original hypothesis that a larger shock area will create a more consistent system by making it easier for the fish to be placed between the panels.

Our tests showed a significantly higher current than we had predicted during our simulations. Using the equation for calculating the resistance of the saltwater, we determined the resistivity value needed for each panel in order to produce the current values we measured. Table 3 shows these values.

Panels	**Resistivity (ρ)**
Square	.082
Circle	.088
Cross	.082

Table 3: Resistivity Values

As discussed previously, the circle panel has a slightly smaller surface area than the calculated value. These calculations were made using the intended surface areas of each panel. Decreasing the circle area in the calculation decreases the resistivity. Therefore, it is accurate to characterize the resistivity values of the saltwater as 0.082Ω*m. This is a little less than half the true resistivity value of saltwater. Temperature as a factor does not explain the difference, as colder temperature makes saltwater more resistive. The only explanation the team could find is in the reaction our panels had with the saltwater. Discussed in Section 3.1.6, the saltwater had a chemical reaction with our stainless steel that caused rust particulates to quickly fill our saltwater basin. The addition of these particles in our testing medium could explain the lower resistivity. Increased sodium levels could also cause low resistivity, however, saltwater cooking was carefully measured by the team, so the particles are a more likely cause.

3.1.11 Small Panel Tests

We used the fourth set of panels, the small panels, in order to view the edges of the field at a greater relative distance than the large panels. However, this series of tests presented the team with a series of problems. The first problem the team discovered was during our current measurements. After adjusting for the slightly less than 4in (3.5in) separation of the panels and the slightly larger 4in^2 area (4.18in^2), we should have measured a current of 9.18A from the small panels. This value was calculated from the 0.082 resistivity value. Instead, we measured a current of 12.6A. This value was repeatedly measured three separate times. This current indicates the resistance encountered by the small panels was 1.96Ω. We should have seen a resistance of 2.69Ω. This 0.7Ω difference could be accounted for by an increase in area, a decrease in panel distance or a decrease in water resistivity. The team adjusted for the area and distance, which leaves the water resistivity as a variable. The three large panels were measured in close time with each other. The small panels required a change of holding equipment, which disturbed the water and mixed up particulates. This could account for the 0.22Ω*m difference in measured resistivity.

The second issue we encountered was with the resolution of our testing with the saltwater panels. The panels were placed at 6in and 10in, however, measurements were taken at 4in, 8in and 12in. Figure 56 shows the position of the small panels in the potential map.

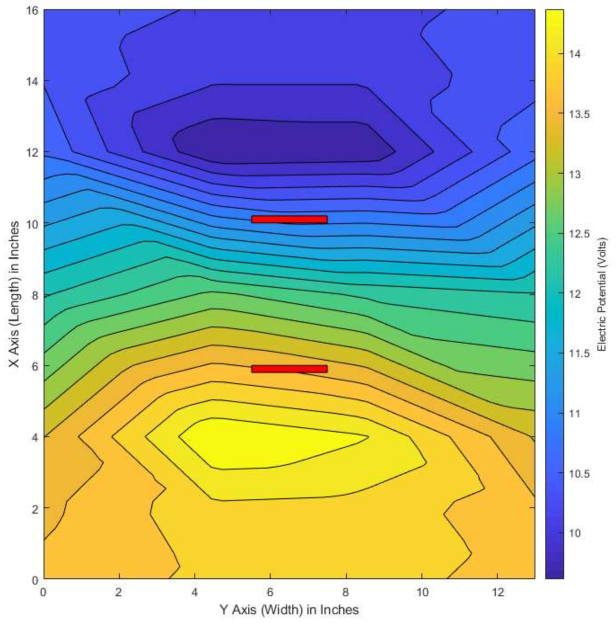


Figure 56: Small Panels – Top-Down – Potential Map

The peaks are located 4in and 12in, however, this is due to the measurement locations. The 20V peak and 0V peak are not present in this plot. They should be located at 4in and 12in, but they have been lost. This is consistent among all the small panel plots. The issue does not affect the applicability of the test since we were interested in the response of the field toward the edges, however, the inconsistency is worth noting. Figure 57 shows a graph with the small panels, with the missing peaks added back in through estimation.

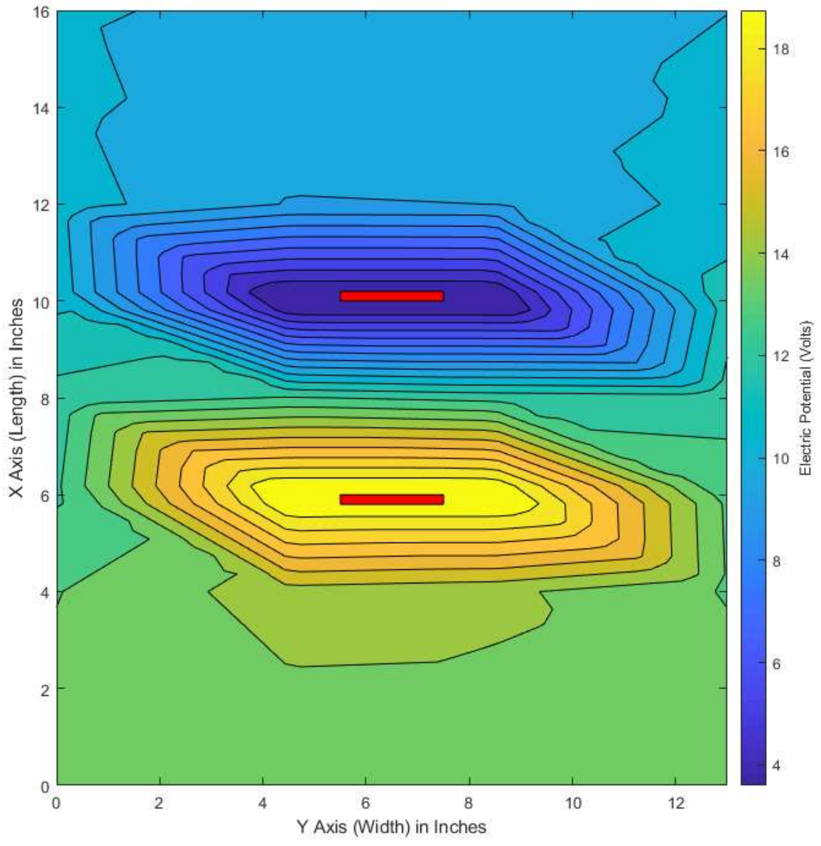


Figure 57: Small Panel – Top-Down – Adjusted Potential Map

3.2 Classification System

3.2.1 Dataset

For developing the dataset, our team gathered images from a variety of resources. Namely, ImageNet, Google Images using the Fatkun image batch plug-in, and images that were gathered by a researcher at North Eastern University working on a similar project. The dataset consisted of 4280 distinct images of lionfish, 1501 distinct images of human divers both scuba and free-divers, 320 distinct images of reef sharks, and 34 distinct images of sea urchins. We decided to include a classification for sea urchins because the physical features of the sea urchin closely resembled those of the lionfish, and we did not want the sea urchin to spoof our object detection system.

One of the reasons why our team used multiple resources for gathering images was so we could get a good variance of images in our dataset. For examples, the images that we received from the researcher at North Eastern were image slices taken from a video of someone hunting lionfish in Florida. These images were useful; however, they were all taken in the same lighting conditions which can make our object detection classifier less robust. So, by gathering images from multiple resources we ensured that we would have a good variance of images in different lighting conditions, profiles, and filtering. Below, we have two images of lionfish in our dataset which showcases the image variance we referred to in the background section of this report.

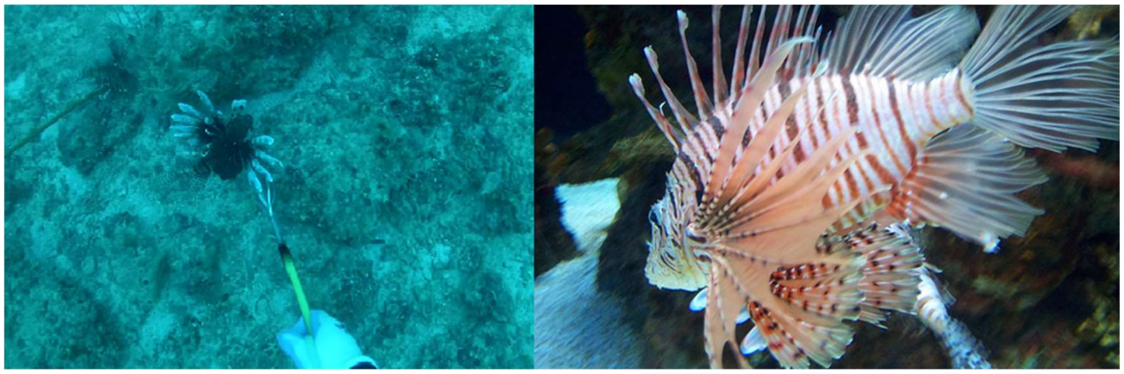


Figure 58: Lionfish Dataset Images

Our team used Dataturks, an online, collaborative crowd-sourcing image annotation tool for annotating our images. Dataturks provides a friendly user-interface and an easy way to distribute the workload required for annotating the thousands of images we needed for this project. Once we finished annotating all the images in the dataset, we next had to convert them to the TFRecord format, a binary file format that is required to use TensorFlow. To do this, our team needed to convert to intermediate dataset formats. We first needed to convert the Dataturks format to PascalVOC using the python script in Appendix B.1. Next, we converted all images to JPEG format since this is the recommended image format for TensorFlow; the script for this can be found in Appendix B.2. We then used a script that would break up our dataset into a training set which consisted of about 5,500 images and a testing set which consisted of about 400 images, this script can be found in Appendix B.3. At this point we have a folder named ‘annotations’, which contains all of the PascalVOC format XML files that describe the bounding boxes for each image, we have a folder named ‘images’ which contains all of the images in our dataset, and we have two CSV files, which have the file names of all the images in our ‘images’ folder to distinguish which images will be used for testing and which will be used for training. Finally, we used another script which used our images, CSV files, and PascalVOC format XML files to generate TFRecords; the script for this can be found in Appendix B.4. At this point, we generated two TFRecord files, one for testing and one for training.

3.2.2 Training a Model

Rather than generating our own object detection model from scratch, our team decided to re-train an existing model. The motivation behind this decision was that creating a model from scratch would have required considerably more development time for research, evaluation, and training. Training the model would have also required an order of magnitude more training images to have accuracy performance near that of the pre-trained models, which have all been trained on millions of images.

Our team trained on three pre-trained models. Specifically, we used Inception V2, MobileNet V2 and MobileNet V1. All these models have been trained on the Common Objects in Context (COCO) dataset, is a large-scale object detection, segmentation, and captioning dataset. COCO contains over 200,000 labeled images, 1.5 million object instances, and 80 object categories. The table below shows the speed vs accuracy of each model that we trained. This testing was done using the MSCOCO evaluation protocol and the actual tests were carried out by TensorFlow researchers, rather than the Lionfish Research team.

Name	**Speed (ms)**	**Accuracy (mAP)**
SSD Inception V2	42	24
SSD MobileNet V1 Quantized	29	18
SSD MobileNet V1 PPN	26	20
SSD MobileNet V1 0.75 Depth Quantized	29	16
SSDLite MobileNet V2	27	22
SSD ResNet 50 FPN	76	35

Table 4: Accuracy vs Speed of pre-trained models

For training the models, the team used recommended training configuration files with some modifications. These configurations files outline how TensorFlow will re-train the model, this is where we can modify the number of training steps, data augmentation, learning rate, batch size and much more.

For the actual training process, the team decided to use Google Cloud Platform (GCP) for distributing the computational load since it provided easy access to highly optimized hardware such as Tensor Processing Units (TPUs) and Graphics Processing Units (GPUs) for accelerated the training the process in machine learning applications. This would reduce the time required to train the models and test them by a considerable amount. In our specific case, our team used a Linux machine running Ubuntu 18.04.02 LTS. In order to train models using Google Cloud Platform, we needed to do some environment configuration on the client machine. First, create a project in the Google Cloud Console and enable billing for that project. The specify product from GCP that we will be using is Cloud Machine Learning Engine to run our training job on Cloud TPUs. ML Engine is Google Cloud's managed platform for TensorFlow, and it simplifies the process of training and serving ML models. Next, we will need to create a Google Cloud Storage bucket to store the training and test data for our model, along with the model checkpoints from our training job. We then followed the tutorial on the GCP website for configuring the GCP CLI utility on our Linux machine so we could interact with the GCP project from our Linux command line.

Once the GCP CLI was configured we next had to upload our training and testing TFRecord files along with our 'pipeline.config' file to the Google Cloud Storage bucket in a folder named 'data'. Finally, we needed to package the Object Detection API, pycocotools, and TF Slim libraries and send them to our storage bucket so TensorFlow will work properly. The commands to do this can be found in Appendix B.5.

Completing this process left us ready to submit our training and evaluation jobs to GCP. It is important to note, that training is different for TPU compatible models and non-compatible models. We were able to monitor the progress of our training using the TensorBoard application.

3.2.3 Raspberry Pi Testing

Once the model had been proven to work on a laptop, we began the process of actually gathering information on the performance of an edge device. In this case that edge device is the Raspberry Pi. While this differs from the board RSE will be using it is a good indication of how an edge device may perform for their applications. In doing our testing we wanted to utilize a variety of configurations in order to determine how certain components may impact the performance.

As mentioned in Section 2.4.3, a method for testing each different configuration could be to utilize the stratified 10-fold cross-validation test. However, this test would require us to retrain each model we were testing 10 different times. As mentioned in Section 3.2.2 the amount of computational power it would take to do that was out of the scope of our capabilities. Furthermore, had we decided to run each training on the Google Clusters, the cost would have been well over two thousand dollars. As such we had to adapt and come up with a different acceptance test method. We especially had to adapt after switching from simply labeling an image to an object detection model that bounded each lionfish in the frame. This is because with an object detection model, you need to be able to localize where on the frame the object actually is and put a bounding box around it. Whereas in image labeling it was just a matter of if that object was present.

With this type of object detection, we were able to utilize a holdout test set that was compared against the model and then TensorBoard returned accuracy characteristics specifically the mAP characteristic defined in section 2.3. This performance characteristic is what we will utilize in order to evaluate the accuracy of the different models. While this was a good start, it was important to utilize our own evaluation of the models by going through a test set of frames and calculating precision and recall. This was done by utilizing 20 seconds (200 frames) of video directly from the RSE robot on a mission. This was done by manually keeping track of true positives, false positives, true negatives and false negatives.

Beyond just accuracy we needed to understand the time trade-off that existed. To do this we ran a video, frame by frame through the model and returned a video with the bounding boxes shown on the objects. While this was executing, we collected the time it took to create the bounding boxes for each frame and then averaged out the processing time per frame to get the time performance characteristics of the configurations. This was the process we used for gathering the statistics, but setting up and running the tests on the Raspberry Pi versus a laptop had additional obstacles.

When transferring over the trained models to the Raspberry Pi, we had to reconvert each of them to the frozen inference graphs to ensure there were no issues between TensorFlow versions. This simply meant converting the saved model graph which can still be trained on to a frozen inference graph which is a serialized graph that cannot be trained anymore. After this conversion we were able to utilize the same video processing script utilized on the laptop with some slight edits. The results are then returned to a .csv file with the average frame rate once the video has finished processing. Additionally, note that the accuracy statistic for mAP are gathered during training on the Google Cluster as those results would not be affected by the Raspberry Pi. Compiling the time, mAP, precision, and recall results of the different models offered better insight into the varying performance characteristics.

3.3 Conveyor Intake Mechanism

After weighing the pros and cons of the different designs, we decided to prototype a conveyor intake system. This was decided on by both our team at WPI and the RSE team due to its uniqueness and ability to be built upon. This left much to be determined in terms of composition and materials of the intake mechanism.

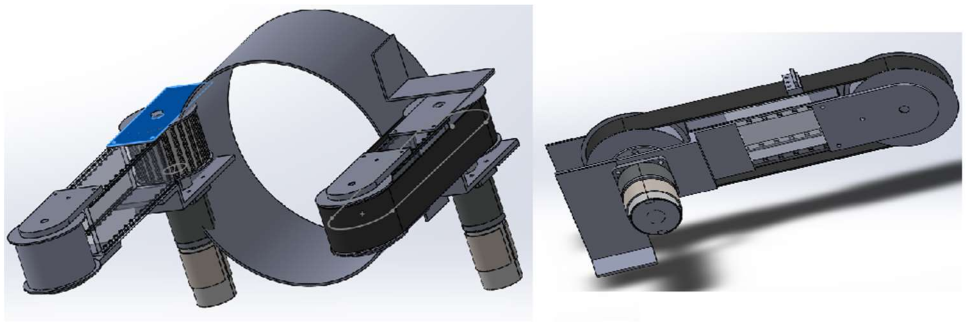


Figure 59: CAD of proposed conveyor

3.3.1 Material selection

As stated in the previous section, a marine environment posed many difficulties in the selection of materials used on the conveyor system. The goal was to create a working prototype that could be used by RSE in future iterations. This led us to make the prototype as genuine to the marine environment as possible in our timeframe and price range. Therefore, many of our parts such as the conveyor body and free spinning roller were made of polycarbonate and stainless steel; both with a low risk of corrosion. However, both the off-the-shelf conveyor belt from Brecoflex and the PG71 motor from AndyMark were made of polyurethane and steel. This was due to price and availability.

3.3.2 Phalange Design

For the phalanges, a very organic finger-like design was used in order to scoop lionfish without harm. Inspiration for this design came from the shape and movement of the sea anemone. Additionality, the length of two phalanges needed to be long enough to stretch the 10-inch diameter of the RSE Guardian LF that contains the lionfish. Figure 60 is the CAD model of what a single finger looked like.

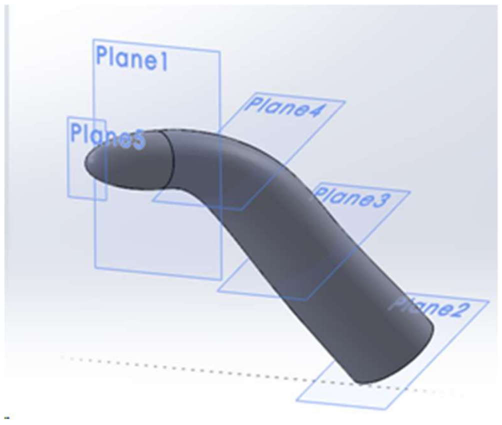


Figure 60: CAD of Finger Design

To mount the phalanges to the conveyor belt, a thicker base was needed so Figure 61 shows a model of what the fingers and base looked like.

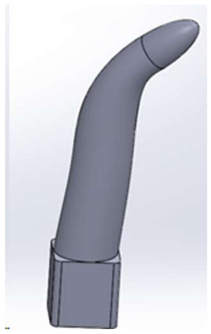


Figure 61: CAD of Finger with Base Design

3.3.2.1 Molding and Material Selection

After creating the design, we needed to determine what specific material we were going to use to fabricate the phalanges. We wanted a material that would be flexible enough to not harm the lionfish, yet stiff enough to prevent it from escaping. We compared possible materials to the Durometer Shore hardness scale to determine whether they had the proper amount of flexibility for our solution. This scale was created to give people a common understanding of different materials hardness with baseline materials. This scale includes Shore 00 for super soft materials, Shore A for medium hardness materials, and Shore D for very stiff materials (Durometer shore hardness scale, 2019). In Figure 62 you can see the reference materials on a scale to determine the hardness of a material relative to common baseline materials.

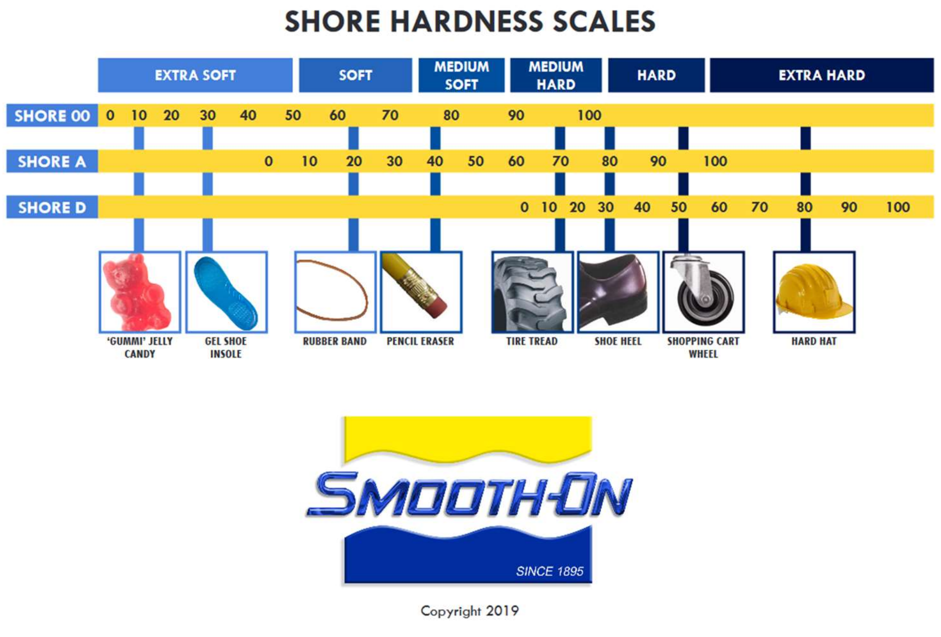


Figure 62: Shore Hardness Scale Material Comparison (Durometer shore hardness scale, 2019)

We immediately decided that we did not want to use a Shore 00 or Shore D materials because they would be too soft or too hard, respectively. With that decided, we visited the Interim Director of the WPI Rapid Prototyping lab, Erica Stults, who had samples of the different Shore

A materials and decided that Shore A 30 had the right flexibility and stiffness for our application. Once we decided on a material hardness level, we then needed to choose a specific material that approximately met that hardness. Mold Star 30 was chosen because it is an aquarium safe silicone material that was readily available online.

After the specific material was chosen, we needed to create the actual mold for creating these oddly shaped and intricate phalanges. After thorough research, we concluded that 3D printing the mold would be sufficient for our purposes. We then took the negative of the phalange design and put it into a rectangle. This model was 3D printed and we then mixed the Mold Star material weighting each part to ensure a proper ratio. The mixture was degassed in a vacuum chamber to remove air bubbles and poured into the greased mold. However, when the time came to release the mold, we experienced some difficulty removing it from the mold because of the vacuum seal created by the mold and material. Eventually, we needed to cut the mold in half to release the mold as seen in Figure 63.

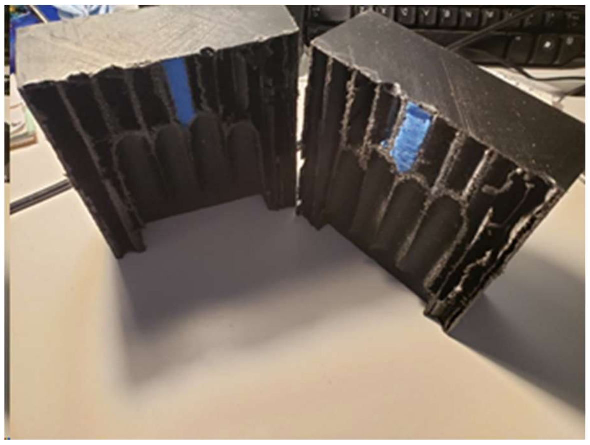


Figure 63: Mold Cut in Half

We redesigned the mold to be a two-part mold which was held together with M3 screws as seen in Figure 64, which allowed for a much easier mold release to produce what we see in Figure 65.

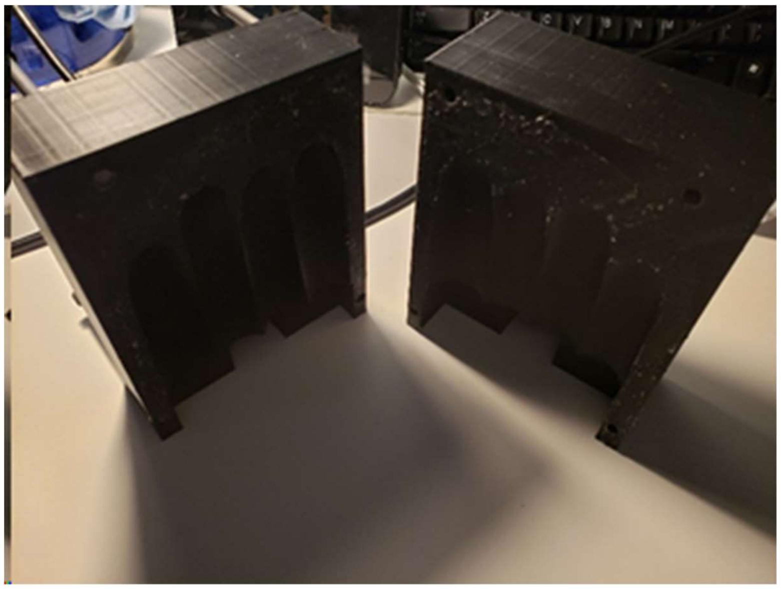


Figure 64: Two-part mold

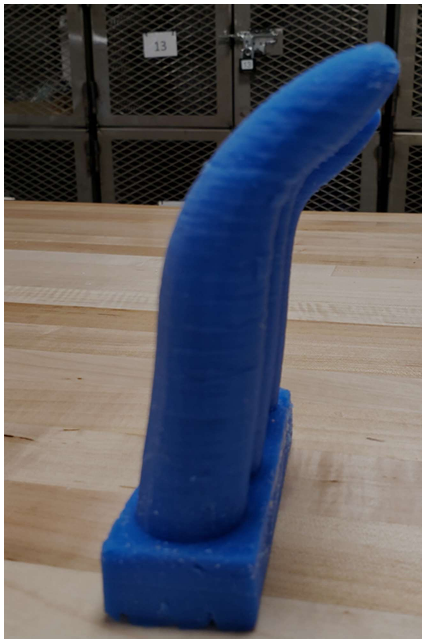
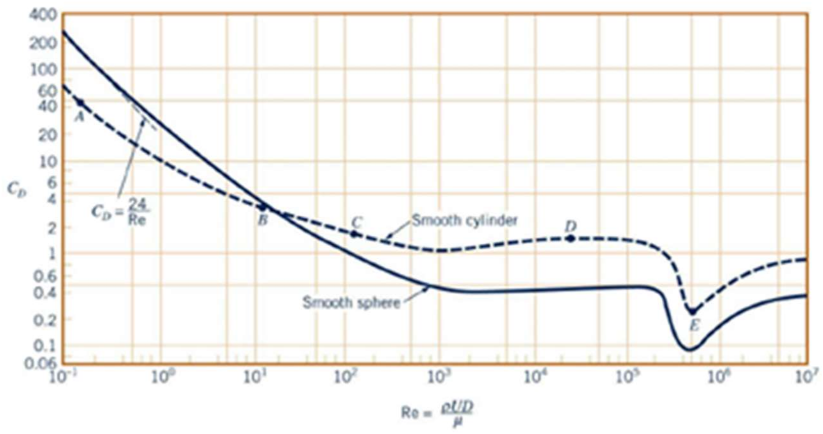


Figure 65: Product of Mold

3.3.3 Drag Calculations for the Intake Mechanism

To determine the requirements for the motor, we needed to understand how drag would affect our system. Drag in our system was caused by the phalanges and lionfish opposing the medium of water which is also known as viscous drag (Granville, 1976). The drag was determined by the shape of the object being dragged through the medium and because our lionfish and phalanges were so oddly shaped, we modeled them as spheres and cylinders respectively. The equations and chart in Figure 66 were used to determine the drag force for each group of phalanges and the drag of the lionfish.



$$F = 0.5 * C_d * \rho * A_{\text{frontal}} * V^2$$
$$C_d = \frac{24}{R_e}$$
$$R_e = \frac{\rho U D}{\mu}$$

Figure 66: Calculations and Chart to determine Drag Force (White, Rhim, 2016)

After completing the calculations, the drag force was determined to be approximately 10 ft-lb for the four phalanges and the lionfish. However, this was a very conservative estimate because calculating the exact drag would be very difficult and this estimate is more than enough for determining the motor needed for this mechanism.

3.3.4 Motor Selection

When we first started designing the intake mechanism, we were given two brushless Blue Robotics T200 underwater thrusters from the RSE team. However, as the project progressed, we realized that these motors would not work for our purpose because they were made for thrust/speed and not torque. As determined in the previous section, we would need at least 10 ft-lb of torque to compensate for the drag caused by the underwater environment. Therefore, we started to look for waterproofed motors with high torque and low rpm. On the market waterproof motors are more focused on thrust rather than torque because applications like ours are very uncommon. This had us thinking outside of the box and led us to the AndyMark PG 71 brushed motor with planetary gearbox as seen in Figure 67.

Figure 67: AndyMark PG 71 Motors (PG series gearboxes)

It has a stall torque of 16.3 ft-lb and 75 RPM no-load which was needed for the system and we always knew what direction it would spin because it was brushed. In a later section, we will discuss how we waterproofed this motor for our application.

3.3.5 Conveyor Belt Design and Selection

The conveyor belt design was one of the most important components of the entire system because it allowed for the mechanism to move to grab the lionfish. Initially, we tried to use polyurethane sheets, however, we found that it was difficult to tension the belt and fasten our phalanges to it because any cut or hole in the belt would cause the belt to completely rip. Additionally, it made attaching the phalanges near impossible because holes were needed to attach them. Therefore, we decided to purchase a commercially made specialty conveyor belt. We purchased a polyurethane and steel reinforced timing belt from Brecoflex that allowed for strategically placed holes to be created without damaging the belt. These holes were then filled with plastic inserts to allow for cleats to be affixed to the surface. A stainless-steel reinforced option was available but was more costly for just a prototype design so it was decided that if the design was pursued RSE could purchase the correct belt reinforcements.

3.3.6 Cleat Design

The cleats provided us with an easy-to-use mechanism for attaching the phalanges to the belt. They were designed to be small and fit with the plastic inserts that allow for screws fit inside for securing as seen in Figure 68.

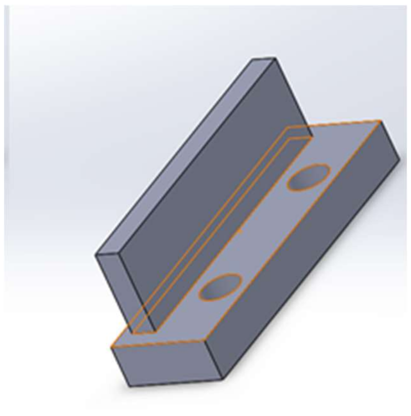


Figure 68: Index Cleat

3.3.7 Roller Design

When originally designing the rollers, we tried to machine small rollers that would be tensioned with the base of the mechanism as seen in Figure 69.

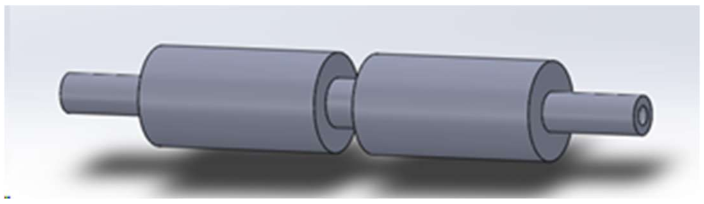


Figure 69: Old Roller CAD

However, we soon found out that tension rollers were difficult to tension and had very little traction with the sheet polyurethane. We decided that a commercially made timing pulley as seen in Figure 70 by Brecoflex was a better solution because the rollers paired with the belt had much more traction.

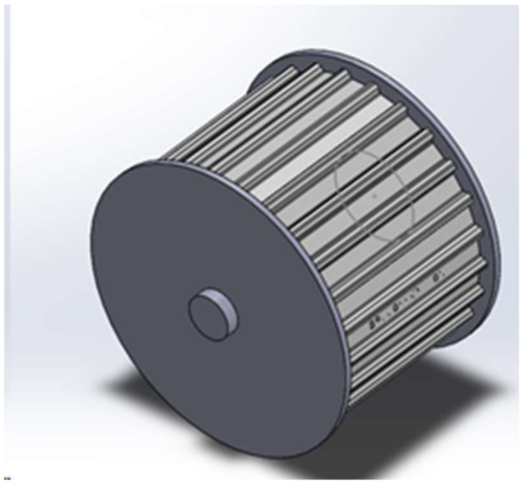


Figure 70: Brecoflex Timer Pulley

The Brecoflex pulley was made of Delrin and steel which would need to be changed for a marine environment application. We also looked into 3D Printing these rollers, which happened to be much less expensive and will not corrode in saltwater. Additionally, it was recommended that only one timing pulley be used per conveyor and that a non-timing pulley is made to avoid the high commercial cost. Figure 71 shows a CAD model and physical machined product created from polycarbonate and stainless steel.

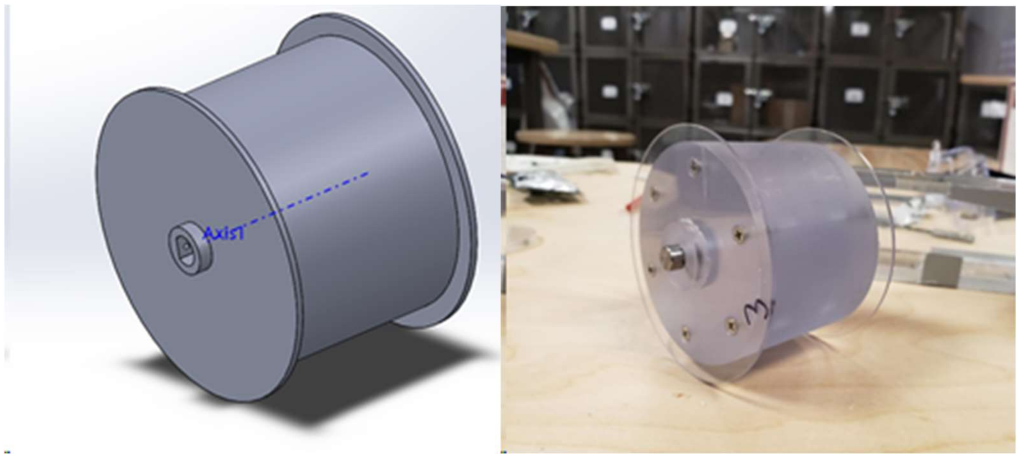


Figure 71: CAD and Physical Free Spinning Roller

3.3.8 Creating an Actuated System

For system ease of use, a semi-autonomous conveyor system was designed so that both conveyors would move to the same position after one full rotation, which is the average length of a lionfish. This would allow for the operator to know that the lionfish was completely ingested into the body of the RSE robot capture container. To do this we utilized a hall effect sensor. A Hall effect sensor works when an electric current flows through a magnetic field and charges one side of a conducting material. When this charge gets large enough it creates a voltage that can be measured (Hall effect sensor and how magnets make it work, 2013). In our case, we used the Hall effect to determine where the belt is placed because when the sensor sees the magnet it will stop the motors from moving. The program flow diagram can be seen in Figure 72.

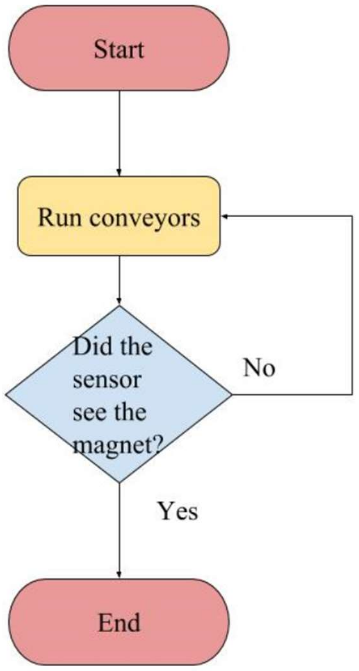


Figure 72: Program Flow Diagram for Hall Effect sensor

3.3.9 Waterproofing Motors and Sensors

We waterproofed the motors by following a RobotShop community post. The technique utilized Sugru – a moldable glue – and required packing grease into the gearbox (Waterproofing a DC motor with Sugru, 2011). By doing this, we created a waterproof seal for the gearbox and patched up all the cracks where water can leak and destroy the motors. These motors were covered in multiple layers of the Sugru which can be seen in Figure 73.

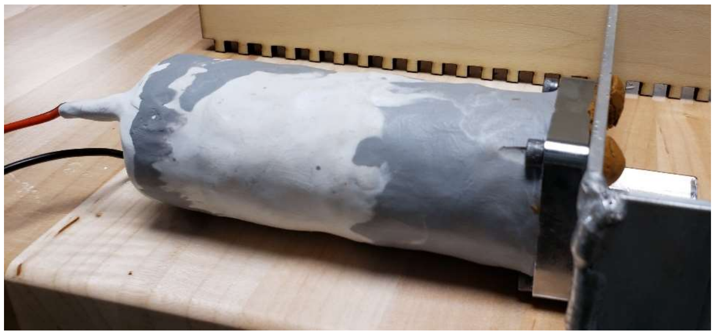


Figure 73: AndyMark PG 71 Motor Encapsulated in Sugru

For our prototype we coated our hall effect sensor in multiple thick coats of silicone sealant, which is an excellent waterproofing material. This sensor was tested in both salt and fresh water and we did not see any significant differences after we coated the sensor. Note that the sealant we used in Figure 74 is only rated for less than 30 gallons tank, due the fact that this was an aquarium tank, however it proved the concept.

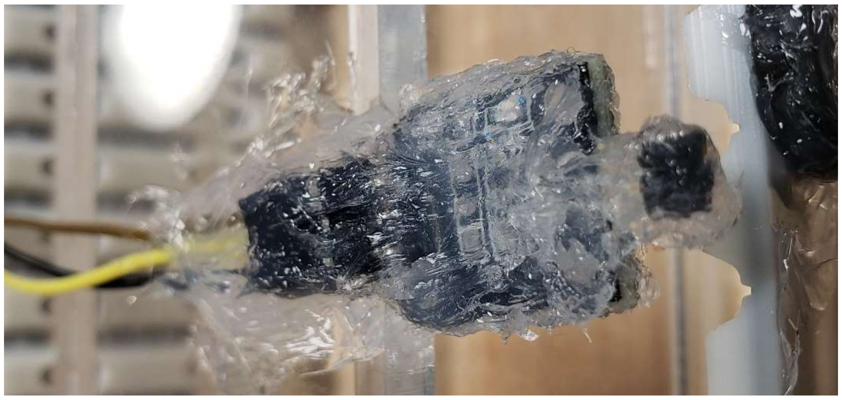


Figure 74: Hall effect Sensor Covered in Silicone Sealant

4. Conclusion and Recommendations

4.1 Electrode System

4.1.1 Field Documentation

The field generated by the panels is equally distributed throughout the nearby salt water. As seen in Figure 75, the field is uniform between the two panels and rounded, similar to the dipole image discovered during our research. The field strength is concentrated between the two panels and characterized by two ovals that are connected in the middle by a narrow segment.

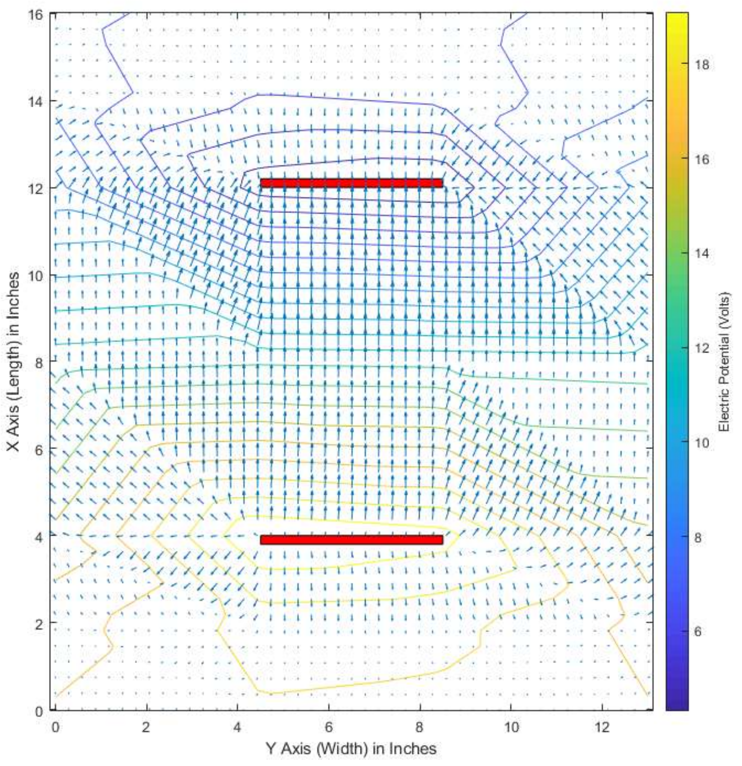


Figure 75: Large Square – Mid-Level – Top Down Field

There was very little noticeable difference between the panels that had the non-conductive material on the back and the panels that did. The mid-level top view layer of a non-conductive panel and a conductive panel are shown in Figure 76 and Figure 77. The distribution of the field is largely identical, however, the panel with non-conductive material does have slightly tighter potential bands. This indicates that covering the panels did affect the field, however, not significantly. Both graphs indicate readings of 16V at the edge of the basin.

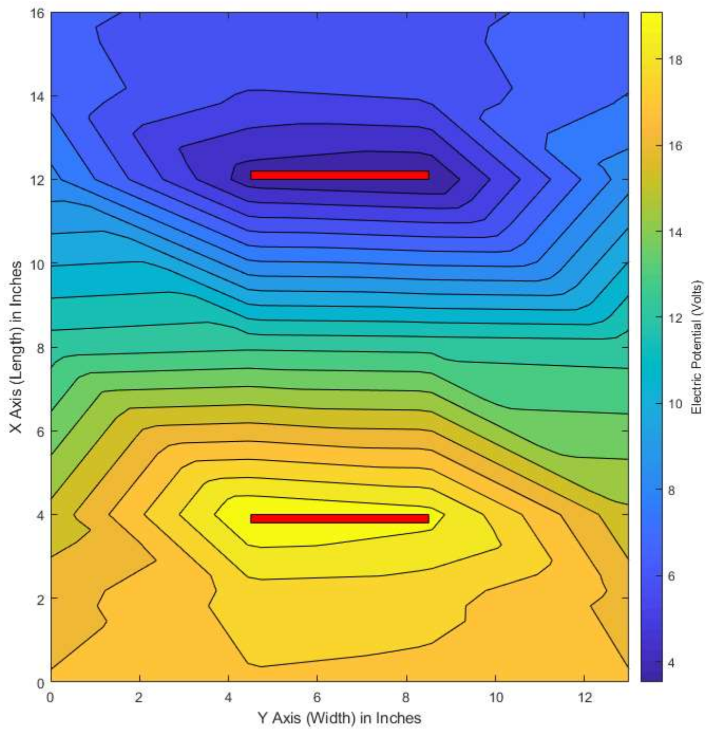


Figure 76: Conductive Backing – Mid-Level – Top Down Potential

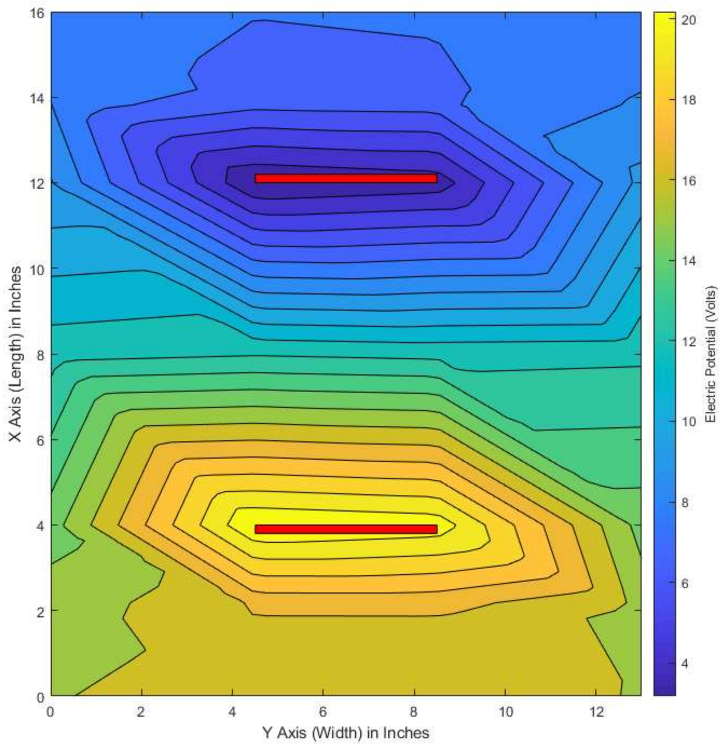


Figure 77: Non-Conductive Backing – Mid-Level – Top Down Potential

The small panels were designed to give us a view of the field shape out to a farther distance than the larger panels. The mid-level top view layer of the small panel is shown in Figure 78. The voltage values as we move farther away from the panels continues to decrease beyond the points. The shape of the field does not change.

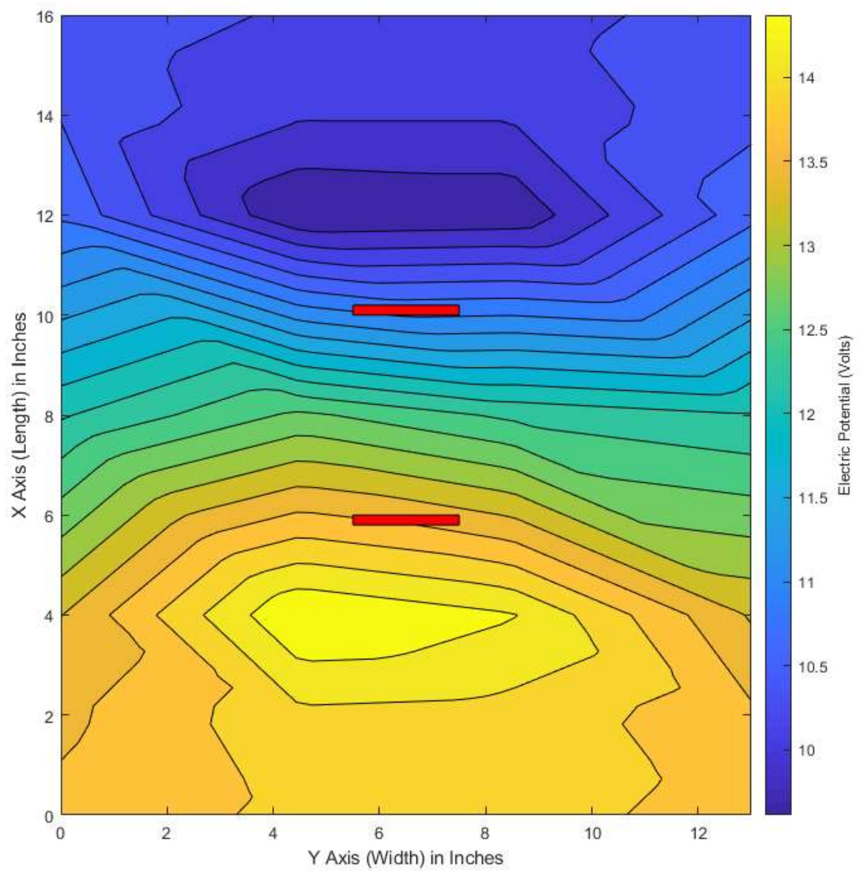


Figure 78: Small Panels – Mid-Level – Top Down Potential

The field generated by the system is consistent with our simulations and hypothesis. It is, however, the electrical current between the panels that generates the system's shocking ability. The system was tested in its steady-state. Further tests should explore the effects on the system by the determined frequency. Such tests should explore the impact of the skin effect on the wire. Further tests to identify the exact current being used by the system could also prove beneficial in determining the optimal voltage peak, pulse width and duty cycle. Finally, a finer resolution on points measured in the saltwater, specifically, the areas immediately surrounding the panels would offer more insight on the voltage across the panels. Our test concentrated on the areas behind and outside the panels.

4.1.2 Panel Shape Recommendation

The team set out to increase the consistency of delivering an effective shock. In order to do this, we looked at a variety of panel shapes. From these tests, we were able to determine the consistent field shape and the uniform current distribution. The problem of inconsistent shock can be tackled from two angles. First, the panels are not delivering an adequate shock due to lack of a high enough current. Second, the panels create an effective shock area that is so small that it is difficult for a user to align the lionfish correctly between the two panels. The first issue is one that was addressed by the RSE team when they increased their system voltage from 24V to 28V. The RSE team noted that they had observed an increase in effective shocks delivered. To the second issue, increasing the size of the panels creates a larger shock area, however, it also decreases the resistance, thereby increasing the supplied current and increasing power draw.

The larger current creates a higher power draw on the system. The team explored the second issue, with the two findings from our field tests. Given the uniform current distribution over a small area and the need to shock a lionfish without physical contact, a larger panel that maintains the original current draw is the optimal solution. Taking a larger panel and covering it with non-conductive material decreases the conductive surface area. The team's recommended panel design is shown in Figure 79. Figure 80 shows representation of a lionfish and the green shows where current would flow.

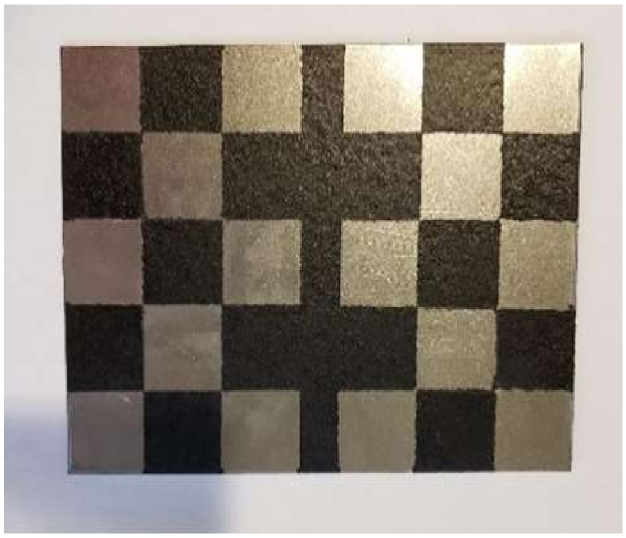


Figure 79: Grid Shape Panel Recommendation

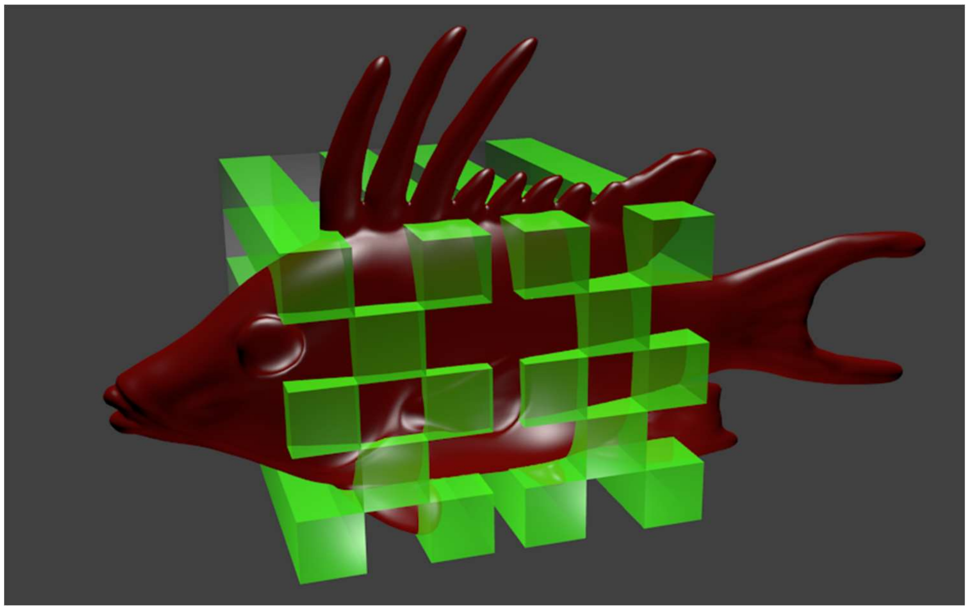


Figure 80: 3D Effective Shock Area

This panel design will maintain the required current for an effective shock but will spread the current paths out over a slightly larger area. This creates a larger shock area which will allow for easier positioning of a lionfish between the two plates. This will, in turn, deliver more consistent results in the field. Future testing of panel design should explore the benefits of non-conductive coating, compared with physical cut-outs of the steel panels. Cut-outs would decrease panel weight and potentially manufacturing costs, by could affect the current distribution. Other shapes that expand the effective shock area in a similar manner could also prove to be beneficial.

4.1.3 Alternative System Designs

The request of the RSE team extended to non-contact shocking mechanisms, however, the team included an additional design in their research. We believed future applications by the RSE team or by WPI teams could benefit from a proof of concept design of a physical shocking mechanism. The team developed a rigid, but potentially retractable element that would deliver a physical shock. The design, shown in Figure 81, is a cylindrical design that has a chamber inside for wiring. The tip of the component has 4 metal balls attached that are charged to a given voltage and ground respectively in an alternating pattern. The design would require physical contact with the lionfish in order to deliver a shock. It would, however, solve the problem of not being able to project the system forward beyond the platform. The system would also make the field properties largely irrelevant, as it delivered the shock through physical contact.

Given the mechanics of Tasers and the design of the RSE team, the system could implement a pulsing frequency designed to disrupt the lionfish's electrical biological signals. The correct pulse could mimic lionfish internal signals and work in the same way a Taser does when it expends the energy in its target. Careful consideration would also need to be given to the saltwater between the electrodes. Future teams would need to ensure the system can deliver a shock through the resistance of the lionfish's scales.

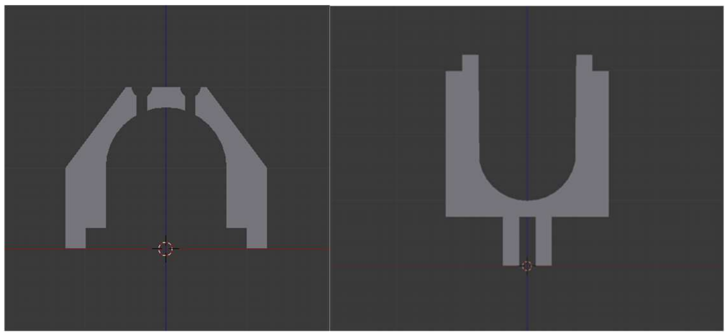


Figure 81: Cross Slice Model Bottom (Left) and Top (Right)

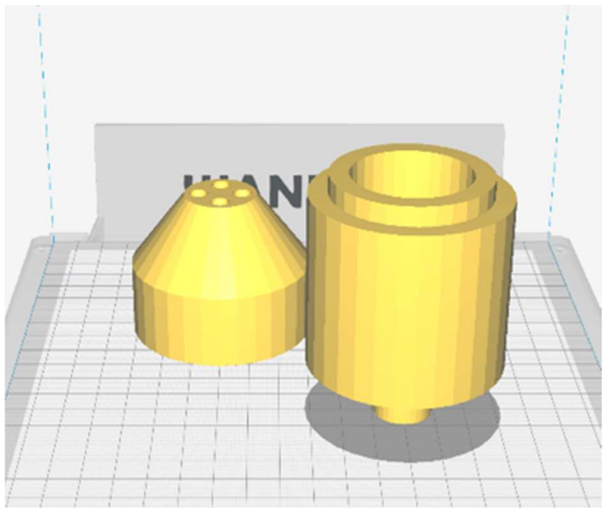


Figure 82: 3D-Modeled Design of Capsule for the Fish-Taser

The team developed a proof of concept which was 3d-printed in two parts. The system utilized 4 - 7/32in ball bearings to act as the electrodes. The 3d-printed component is shown in Figure 83 with the ball bearings attached. Should this design be used in the future, obvious areas of improvement include the width of the element, the material, the waterproof ability and the connection to a retractable rod. The design is currently quite bulky and does not make the best use of space. The component is also not currently waterproof and is held together through pressure. The future design would benefit from a thread-screw style attachment and a waterproof O-ring. The component was also created with PLA plastic the team could easily obtain. A more durable material will be needed in the future.

This proof of concept attempts to address some of the issues we were presented to the team at the beginning of the project. There are several rounds of tests that need to be conducted on the component in order to prove its feasibility. Special attention needs to be given electrode themselves to ensure they can successfully shock the fish. These notes notwithstanding, the proof of concept would solve the range problem and ensure that a shock was delivered consistently to the target.

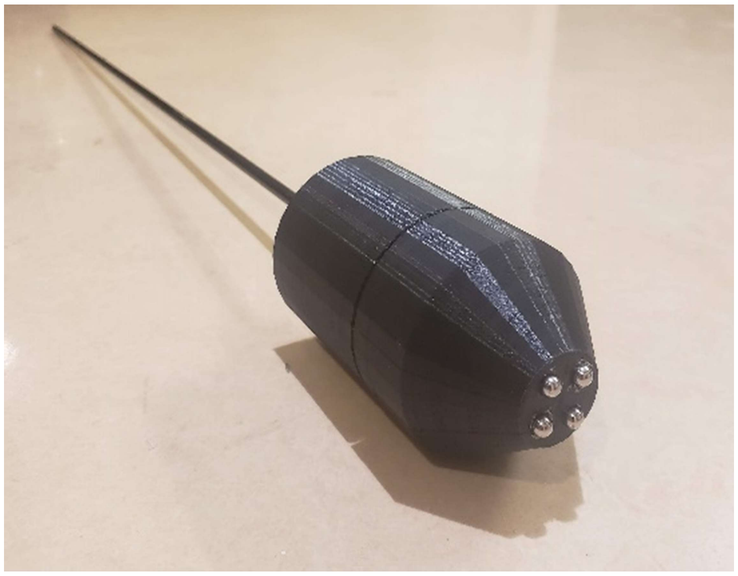


Figure 83: Completed Fish-Taser Proof of Concept

4.2 Classification System

After completing the testing, we compiled the precision and recall characteristics in order to identify some benefits and drawbacks to the different neural networks. We then cross referenced these values with the training output from TensorFlow which provided the mAP characteristics for the top recall performer, ssdlite mobilenet v2, based on the video segment and models we evaluated. The mAP result can be seen below in Figure 84, this graphic was provided as an output from the model training script. According to the figure, after the last step of training the model had a mAP score of approximately 0.315 at 0.75 intersect of union threshold. This information however was only available for this model and doesn't provide a good way to compare against the other CNNs we utilized.

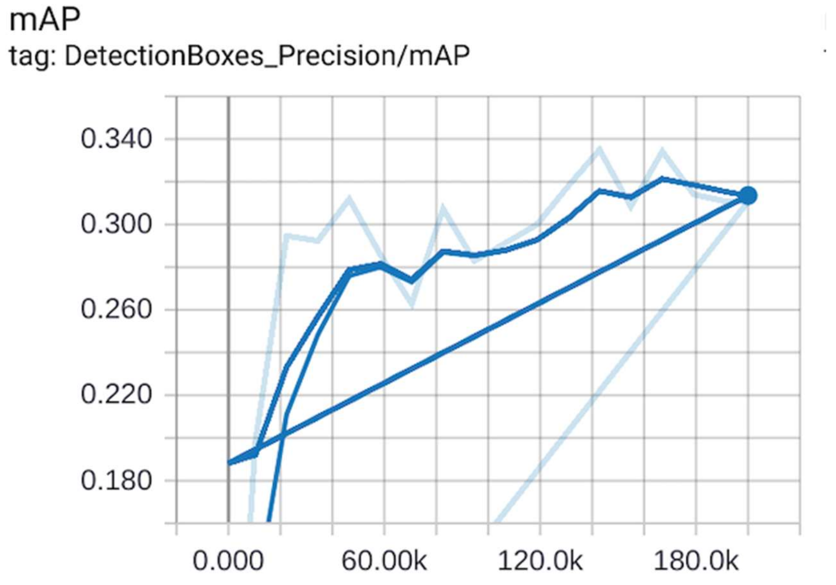


Figure 84: mAP characteristic for SSDLite MobileNet v2 vs steps trained

To do this, we gathered our own precision and recall results on a decision threshold of .75. We extracted a 20 second portion of a video which was shot using the RSE robot camera during a field test in the Spring of 2019. We then took that 20 second video clip and processed it through each individual model, which gave us four different videos. Finally, we went frame-by-frame (roughly 200 frames) through each video and recorded the number of true-positives, false-positives, true-negatives, and false-negatives of classification for each frame. The precision for each of the tested models on lionfish was 1 as there were no false positives when the positive identification threshold was set to 0.75. However, the recall score did vary amongst the different models.

The recall results can be seen in Figure 84 below, with the recall values graphed against the time it took to run a single frame. As you can see, based on the video segment we utilized, the ssdlite mobilenet v2 model had the highest recall score, positively identifying a lionfish 71 percent of the time one was present in the frame. While these statistical results point to clear advantages in our application for the ssdlite mobilenet v2, there are some objective observations worth noting.

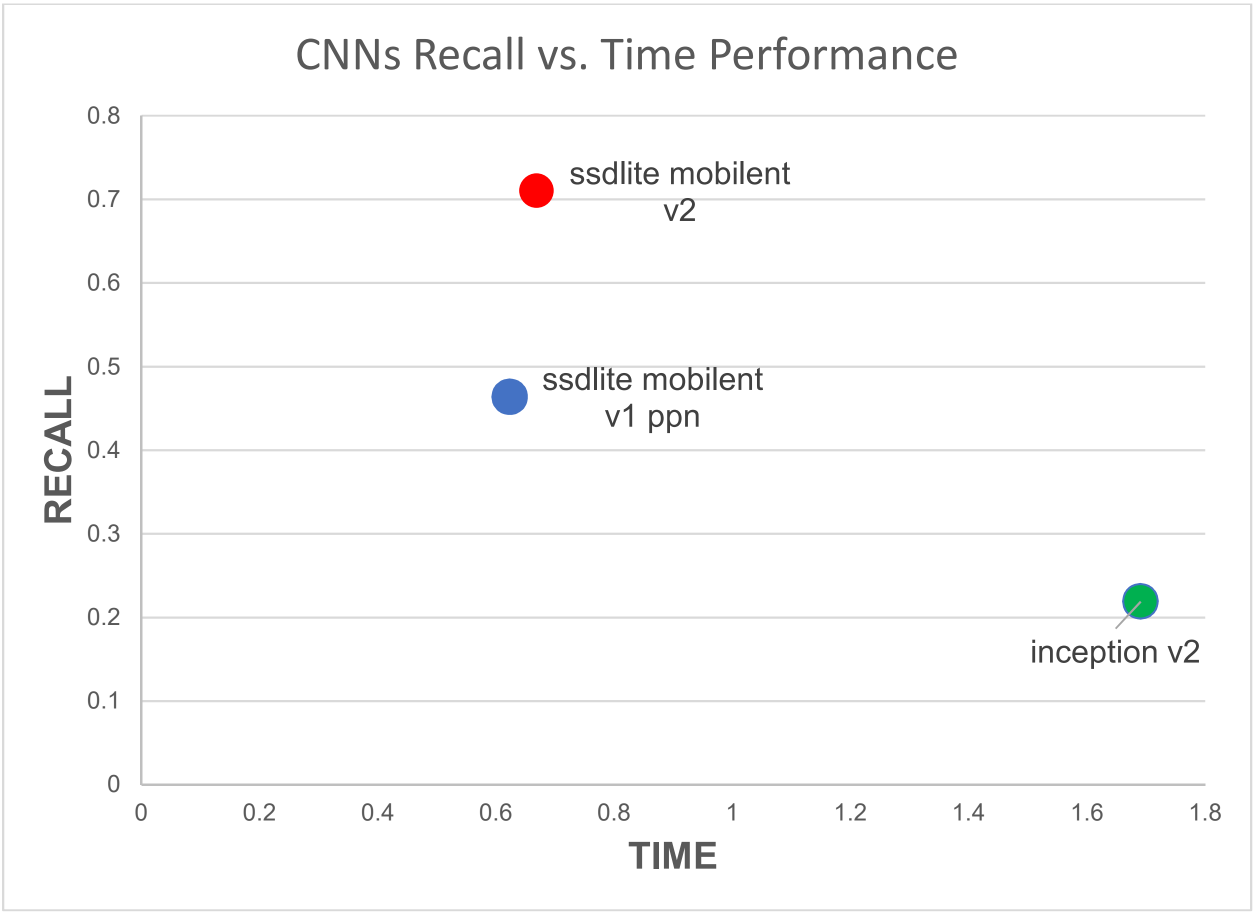


Figure 85: Recall vs Time Performance

For one, during additional testing with a separate video from the one mentioned above from the RSE robot the inception v2 model did identify approximately 10 frames of false positives for human divers and 3 for lionfish. The ssdlite mobilenet v1 ppn model had issues with upwards of 50 false positives for human divers in the given 20 second clip, however we were looking at lionfish classifications for our precision and recall scores. However, this consistently appeared on the right electrode arm in the frame and we believe the classification system mistook the panel and arm in the frame for some portion of a human diver. While the inception v2 recall score was surprisingly low we did notice that when it did identify a lionfish it was normally >95% confident in that decision. In further testing we noticed that when the robot sees the lionfish from a downward looking angle it seemed that the recall was objectively much better. Lastly, we elected not to graph the ssdlite_mobilenet_v2_0.75_depth_quantized model. This is because for the video segment we used for testing, the recall value was zero. It wasn't until after midway through a different portion of video provided by RSE that the model identified a lionfish for the first time. While it did more consistently identify lionfish in this video portion, objectively because the angle of the camera on the lionfish was different, it was not the same video segment the other models were run on so we couldn't change our results. With this in mind, we elected to not include the model in Figure 85 as it would have only had a recall score of zero. However, all of the raw data is provided in an excel document with the model files.

With the combination of these statistical and objective observations we will be recommending the RSE team utilize the ssdlite mobilenet v2 model. We also recommend that next year's team potentially utilize the combined results from this project and last year's MQP in order to implement a solution which utilizes a larger and more diverse dataset, the ssdlite mobilenet v2 model, and the Intel Movidius Neural Compute Stick in order to power the computer vision basic operations. However, once the team moves towards adding further navigation capabilities it may be worth adding classes to the model to allow the navigation to do more than identify and move according to the location of a lionfish in a frame.

4.3 Conveyor Intake System

4.3.1 Building the Intake Mechanism

After we finished the design process of the intake mechanism, we started constructing the prototype by referencing the CAD design as shown in Figure 86.

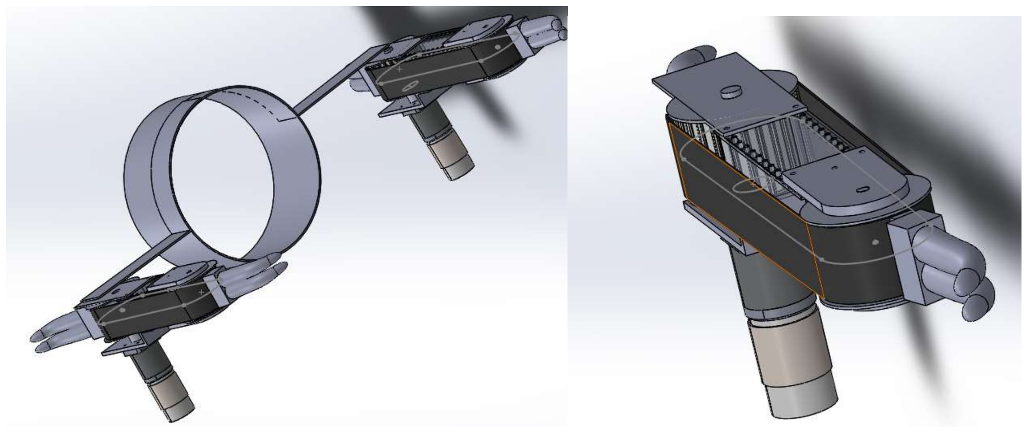


Figure 86: Left: CAD Design of possible Conveyor Belt arrangement, Right: Conveyor belt CAD Design with phalanges

We started building by laser cutting the Lexan to the right size for the width sizing of the powered indexed and free spinning rollers. We taped the laser cut pieces in place to ensure more accurate rectangles then screwed in corner brackets in the corners to keep the pieces in place as seen in Figure 87.

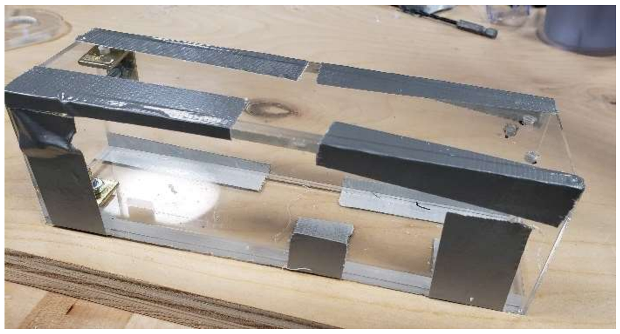


Figure 87: Laser Cut acrylic pieces set for drilling and corner braces

From there we took the free spinning rollers, which were fabricated by Tom Partington in Goddard Machine shop, seen in Figure 88, and aligned them with the Lexan roller holders.

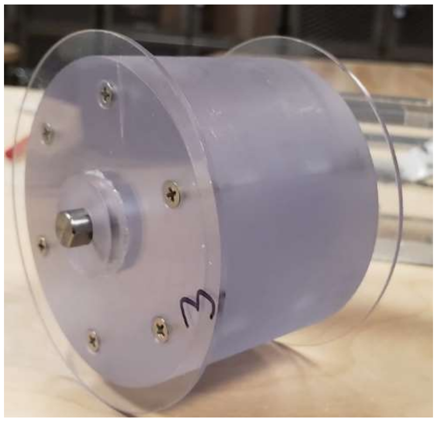


Figure 88: Machined free spinning roller created by Tom Partington in the Goddard machine shop

Marine safe grease was applied to reduce friction between the roller face and holder. While building the base, we also modified the conveyor belt by adding the plastic inserts to the places we wanted to put the indexing cleat. A slit was cut into the phalanges to fit the index cleat, which was sealed with silicone seen in Figure 89.

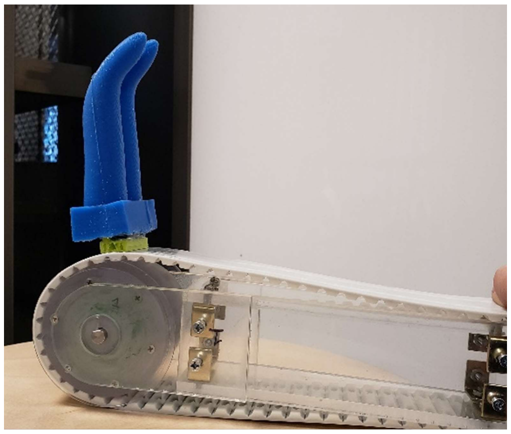


Figure 89: Conveyor Belt attached to Assembled Mechanism

We made sure to place the two phalanges in different directions so that they would actuate properly when spinning. The powered roller and its holder were subsequently attached along with the fully dry conveyor belt with phalanges. Lastly, the motor and bracket were attached along with the sensors and magnets to complete the design which can be seen in Figure 90.

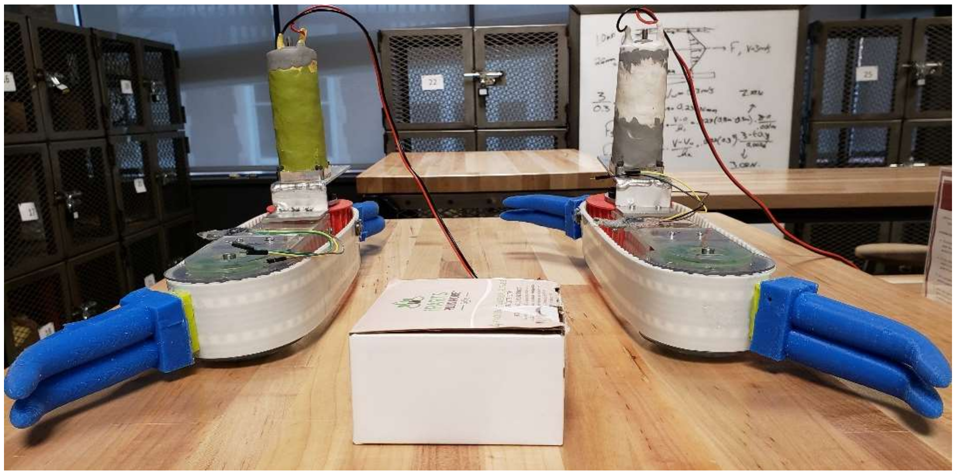


Figure 90: Both the left and right conveyor assembled and ready for testing

4.3.2 Intake Mechanism Prototype Testing

Testing was a very important component in determining whether the proposed prototype would be able to work in our proposed environment. Therefore, tests in and out of the water were performed to ensure it was in proper working order. In Figure 91, you can see the conveyor belts being tested at a laboratory bench in the Foisie Innovation Studio.

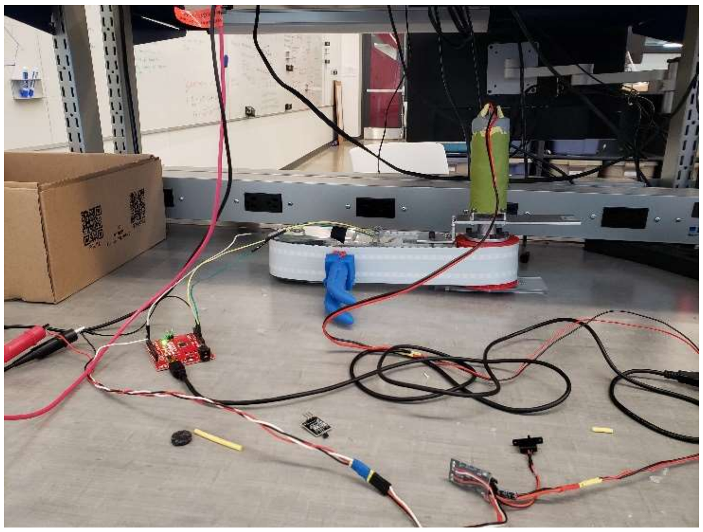


Figure 91: Testing of single conveyor in the Foisie Robotics Lab

We wired the tested sensors, motors, and motor electronic speed controllers (ESC's) to the Arduino Uno that we decided to use because of ease of access. The mechanisms were then powered via a power supply set to 12V and 2 Amps. The waterproofed motor spun at approximately 32 RPM or about .4 ft/sec at full speed from the ESC. Figure 92 is a picture of both index mechanism in-taking a rubber fish in the lab.

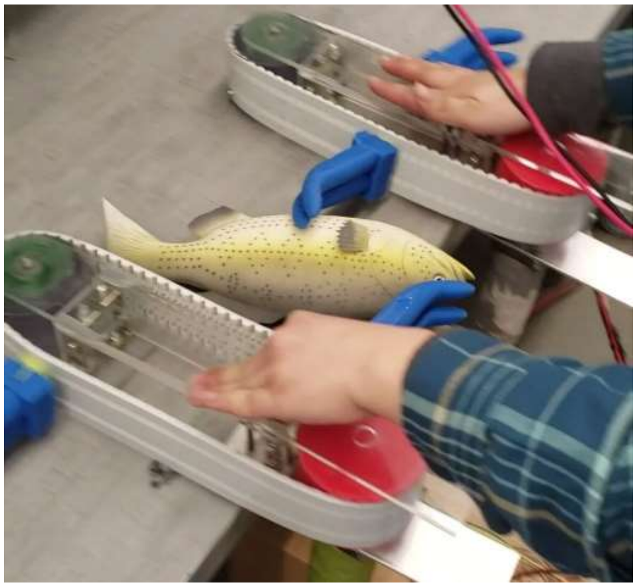


Figure 92: Intake mechanism working to intake a rubber fish to showcase to demonstrate functionality

It worked very well, and the hall effect sensor worked very well. Next, we decided to test in a water environment since that's where this mechanism would eventually end up. This intake mechanism eventually wound up in the bathtub for testing with quite a bit of water seen in Figure 93.

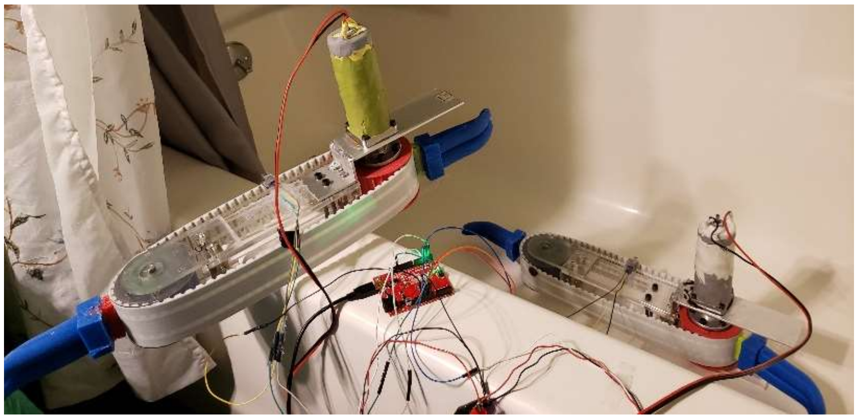


Figure 93: Secondary Test in bathtub to determine the intake mechanism's efficacy in a water environment.

It worked well but we were not able to fit both conveyors in the tub because of their size. Additionally, because we were constantly moving the parts from place to place and storing them in tight quarters, the hall effect sensors kept getting bent and sometimes breaking. We ended up replacing both half-way through testing. To conclude our tests, we purchased a small kiddie pool to test both conveyors underwater, which can be seen in Figure 94.

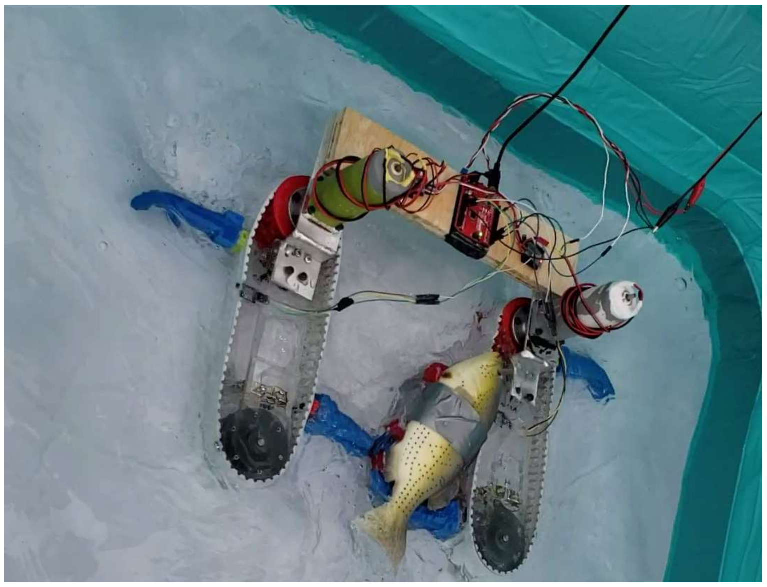


Figure 94: Intake Mechanism Collecting Weighted Fish While in Water Filled Kiddie Pool

The pool was filled with a large amount of water and tested with a weighted rubber fish which it was able to capture quite well. However, after much testing and moving of the parts from place to place, we damaged the replaced hall effect sensors again leaving them much less effective. This made indexing in the pool inconsistent at best, but it proved its effectiveness is larger amounts of water with both indexing mechanisms. We concluded that this design would be effective for the RSE team and it can be the current RSE base CAD model.

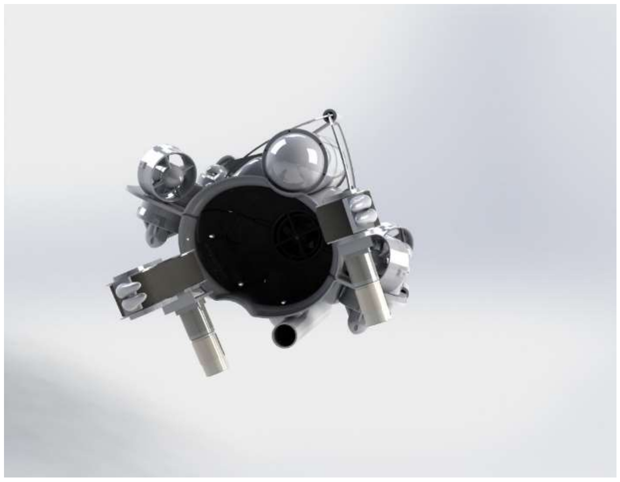


Figure 95: Final CAD of the RSE Guardian with Conveyors attached

4.3.3 Indexing system

4.3.3.1 Electrical

To index the actual system, the motors needed to be correctly equipped with the proper motor controllers and the hall effect sensors need to be connected. This required us to determine the wiring of the Arduino Uno board with the motors and sensors which would enable the actual indexing. To control the 12V DC motors, we borrowed two DC motor ESCs from Keven Harrington from the Foisie Robotics Lab. The ESC has a signal wire that then goes back to the Arduino. The final wiring diagram can be seen in Figure 96.

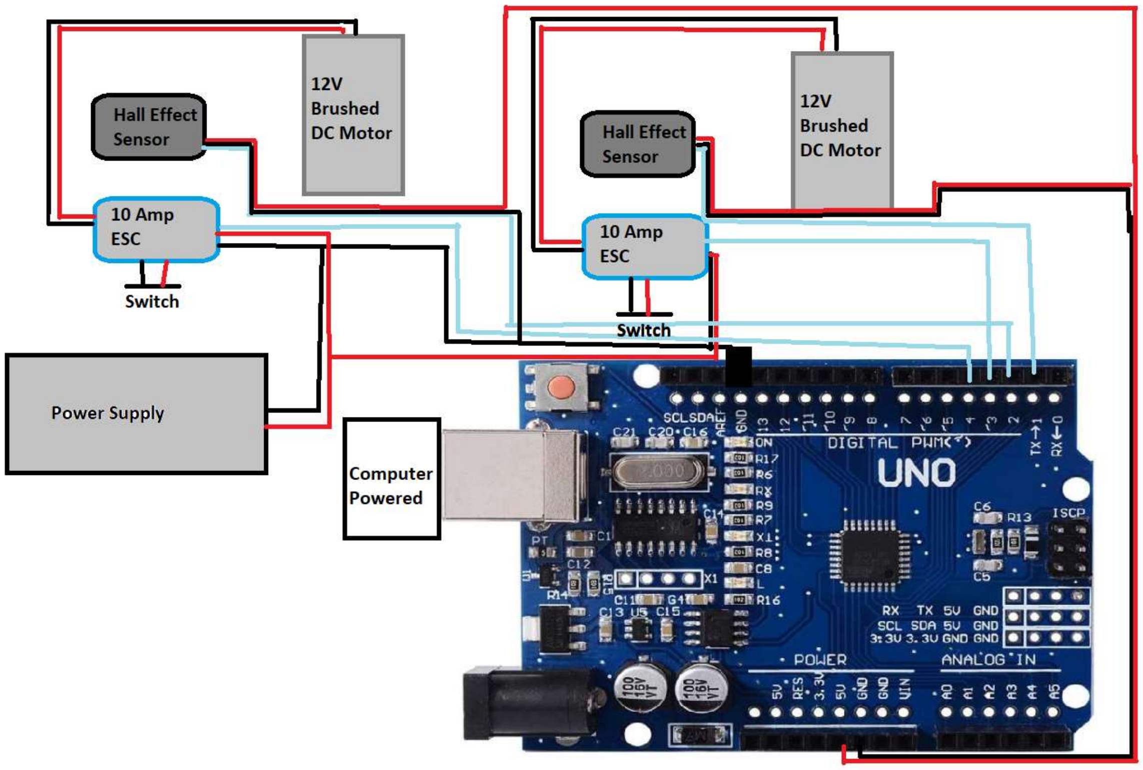
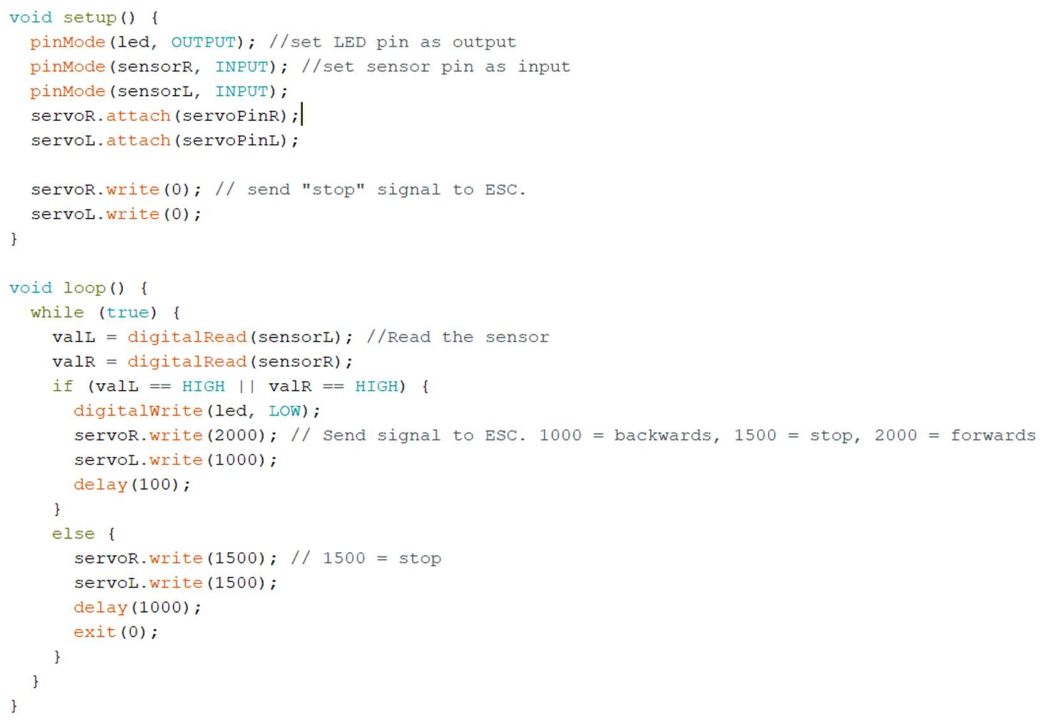


Figure 96: Electrical Diagram of the intake mechanism's motors, sensor, and Arduino connections

We then waterproofed the wires to ensure no water was not able to get into the connection. As stated earlier, we did find we had a rough time with the longevity of the hall effect sensors due to prolonged poor storage. However, it worked fairly well and could be improved upon in the future.

4.3.3.2 Control System

The intake mechanism was actuated through a closed loop control system as described earlier in Figure 72, where the hall effect sensor allows the belt to stop after an allotted distance. On the belt, there were two phalanges which were separated 16 inches apart to account for the maximum length of a lionfish. Therefore, to intake one lionfish at a time we wanted to the mechanism to complete one full rotation before stopping. The hall effect sensors were the solution for a noncontact on/off switch. The code we wrote continuously rotated both motors in opposite directions while reading the hall effect sensors to determine when to stop. When the sensors picked up the magnets, it would talk a little time for the motor to stop.


```
void setup() {
  pinMode(led, OUTPUT); //set LED pin as output
  pinMode(sensorR, INPUT); //set sensor pin as input
  pinMode(sensorL, INPUT);
  servoR.attach(servoPinR);
  servoL.attach(servoPinL);

  servoR.write(0); // send "stop" signal to ESC.
  servoL.write(0);
}

void loop() {
  while (true) {
    valL = digitalRead(sensorL); //Read the sensor
    valR = digitalRead(sensorR);
    if (valL == HIGH || valR == HIGH) {
      digitalWrite(led, LOW);
      servoR.write(2000); // Send signal to ESC. 1000 = backwards, 1500 = stop, 2000 = forwards
      servoL.write(1000);
      delay(100);
    }
    else {
      servoR.write(1500); // 1500 = stop
      servoL.write(1500);
      delay(1000);
      exit(0);
    }
  }
}
```

Figure 97: Arduino Code that Indexes the Two Conveyor Belts

When testing, we learned that covering the neodymium magnets with silicone affected their strength in terms of being sensed, so to attach the magnets on the belt, silicone was applied to the bottoms of the magnets.

4.3.4 Recommendations for Improving the Current Prototype

At the conclusion of the intake mechanism's build and testing, we determined several suggestions to take in mind for the transition from prototype to design that could be utilized by RSE. As stated previously, this is only a prototype so there are aspects that would need to be altered due to our material availability and testing experiences. Since this system needs to be used in deep depths, better waterproofing overall would need to be done. The specific liquid silicone that was used can only be tested up to 30 gallons, this was an aquarium grade silicone, so actual waterproofed hall effect sensors would need to be used to utilize this sensing mechanism. In general, due to the setup of the project, the wiring of the ESC's, sensors, and Arduino was quite messy so a watertight electronics box would help the device significantly. The motors sealed with Sugru held up well in freshwater, however there are many concerns for when the motor is submerged to the depths that RSE hopes to achieve. The deeper the motor is submerged the higher the chance there is for a leak from a tiny crack in the Sugru in the motor. The high pressure will emphasize the smallest cracks and salt water will cause significant damage to the motor.

The mechanism was also not completely created to function in a saltwater environment due to financial and feasibility constraints. For many of our parts we did take the saltwater environment into account, however not every feature was ready for use in saltwater. The constraints that caused this included the price of stainless-steel conveyor belts and the lack of stainless-steel hardware at the local home improvement store. For our testing in freshwater, steel and aluminum would work just fine but in a real-world application these parts would corrode after several diving missions.

Another concern was the weight of the two mechanisms. They both weighed quite a lot due to the motors and the thickness of acrylic sheeting we picked. The motor itself had quite a bit of weight due to its gearbox setup and the fact is was made of steel. Additionally, the free spinning roller was made of solid polycarbonate, which is quite heavy. The weight of the mechanism could be decreased by utilizing 3D printed free spinning rollers and the acrylic pieces with a heavy infill for support in addition to custom or finding a motor more equipped for this environment. With all of these changes, the intake mechanism would be ready for preliminary field testing in the robot's actual environment.

5. References

304 stainless steel in seawater. (2017, -12-20). Retrieved from https://www.clintonaluminum.com/304-stainless-steel-in-seawater/

AbuNaser, A., Doush, I. A., Mansour, N., & Alshattnawi, S. (2015). Underwater Image Enhancement Using Particle Swarm Optimization. *Journal of Intelligent Systems*, *24*(1). https://doi.org/10.1515/jisys-2014-0012

Akpan, N., & Ehrichs, M. (2016, August 24). How do you stop invasive lionfish? Maybe with a robotic zapper. Retrieved October 8, 2018, from https://www.pbs.org/newshour/science/robot-lionfish-invasive-species-rise-nekton

Arlen, T. C. (2018, March 1). Understanding the mAP Evaluation Metric for Object Detection. Retrieved April 15, 2019, from Medium website: https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3

Ataiiyan, Sara. *Fringe Field of Parallel Plate Capacitor*. p. 8.

Bourke, P. (2001, April). Eggs, melons, and peanuts - Cassini Oval. Retrieved from http://lemur.cmp.uea.ac.uk/Research/ivis/backup/PhD/Latest/Paul Bourke Screenshots/Eggs, melons, and peanuts - Cassini Oval.htm

Bowerman, M. (2016, May 26). Whole Foods in Fla. begin selling invasive lionfish. Retrieved October 8, 2018, from https://www.usatoday.com/story/money/nation-now/2016/05/26/florida-whole-foods-invasive-lionfish-food/84962286/

"Caribbean Sea | Sea, Atlantic Ocean." *Encyclopedia Britannica*, https://www.britannica.com/place/Caribbean-Sea. Accessed 11 Jan. 2019.

D'Almeida, W. (2018, January 04). Transfer Learning: Retraining Inception V3 for custom image classification. Retrieved December 14, 2018, from https://becominghuman.ai/transfer-learning-retraining-inception-v3-for-custom-image-classification-2820f653c557

Durometer shore hardness scale. Retrieved from https://www.smooth-on.com/page/durometer-shore-hardness-scale/

Did I Just Catch an Invasive Species? (2016, October 14). Retrieved from https://www.capecod.com/lifestyle/did-i-just-catch-an-invasive-species/

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2013). DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. Retrieved December 14, 2018, from https://arxiv.org/abs/1310.1531.

"ELECTROFISHING." *Veins of Watershed Society*, 2009, http://www.salishsea.ca/resources/Inventory/Fish%20Capture/electrofishing.htm.

Ellison, W., et al. "New Permittivity Measurements of Seawater." *Radio Science*, vol. 33, no. 3, May 1998, pp. 639–48. *Crossref*, doi:10.1029/97RS02223.

Fish and Wildlife Service Occupational Safety and Health. US Fish and Wildlife Service, 29 Dec. 2016, https://www.fws.gov/policy/241fw6.pdf.

Granville, P. (1976). ELEMENTS OF THE DRAG OF UNDERWATER BODIES. Retrieve March 18, 2019, from https://apps.dtic.mil/dtic/tr/fulltext/u2/a031995.pdf

Gerhart, Philip M., et al. *Munson, Young, and Okiishis Fundamentals of Fluid Mechanics*. 8th Edition ed., Wiley Custom Learning Solutions, 2016.

Green SJ, Akins JL, Maljković A, Côté IM (2012) Invasive Lionfish Drive Atlantic Coral Reef Fish Declines. PLOS ONE 7(3): e32596.https://doi.org/10.1371/journal.pone.0032596

Godsey W, Kelly B, Lombardi J, Uvarov N, Yuzvik A. (2018). *Autonomous lionfish harvester.* (). Retrieved from https://web.wpi.edu/Pubs/E-project/Available/E-project-042518-113531/unrestricted/Final_Lionfish_MQP_Paper.pdf

Hall effect sensor and how magnets make it works. (2013, -08-13T15:33:27Z). Retrieved from https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html

Harris, Tom. "How Stun Guns Work." *HowStuffWorks*, 29 Aug. 2001, https://electronics.howstuffworks.com/gadgets/other-gadgets/stun-gun.htm.

Hasted, J.B., Ritson, D.M., Collie, C.H., "Dielectric Properties of Aqueous Ionic Solutions. Parts I and II." Journal of Chemical Physics 16, 1 (1948). http://dx.doi.org/10.1063/1.1746645

How to Retrain an Image Classifier for New Categories | TensorFlow Hub | TensorFlow. (n.d.). Retrieved December 01, 2018, from https://www.tensorflow.org/hub/tutorials/image_retraining#training

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., . . . Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. Retrieved December 14, 2018, from https://arxiv.org/abs/1704.04861.

ICT Monitor Worldwide. (2017). Google Launches TensorFlow Lite for Mobile Machine Learning. *ICT Monitor Worldwide*.

Intel. (2018). Intel® Movidius^{TM} Neural Compute Stick. Retrieved December 13, 2018, from https://software.intel.com/en-us/movidius-ncs

Kızrak, A. (2018, March 4). A Brief Guide to Intel Movidius Neural Compute Stick with Raspberry Pi 3. Retrieved December 13, 2018, from https://medium.com/deep-learning-turkey/a-brief-guide-to-intel-movidius-neural-compute-stick-with-raspberry-pi-3-f60bf7683d40

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *International Joint Conference on Artifical Intelligence*. Retrieved from http://web.cs.iastate.edu/~jtian/cs573/Papers/Kohavi-IJCAI-95.pdf

Li, Y., Lu, H., Li, J., Li, X., Li, Y., & Serikawa, S. (2016). Underwater image de-scattering and classification by deep neural network. *Computers & Electrical Engineering*, *54*, 68–77. https://doi.org/10.1016/j.compeleceng.2016.08.008

Lieber-Kotz, O. (2017, March). Marine Sanctuary Scientist Steve Gittings Fights Invasive Lionfish. Retrieved October 8, 2018, from http://sanctuaries.noaa.gov/news/feb17/sanctuary-scientist-fights-invasive-lionfish.html

Lionfish University. (2018). Trap Research. Retrieved October 8, 2018, from https://www.lionfishuniversity.org/trap-research

Ltd, Copyright Global Sea Temperatures-A. Connect. *Caribbean Sea | Sea Temperatures*. http://www.seatemperature.org/caribbean-sea. Accessed 11 Jan. 2019.

Mackenzie, Fred, et al. “Seawater.” *Encyclopedia Britannica*, 2011, https://www.britannica.com/science/seawater.

Meissner, T., and F. J. Wentz. “The Complex Dielectric Constant of Pure and Sea Water from Microwave Satellite Observations.” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, no. 9, Sept. 2004, pp. 1836–49. *IEEE Xplore*, doi:10.1109/TGRS.2004.831888.

Morell Apr, V. (2017, December 11). Mystery of the Lionfish: Don't Blame Hurricane Andrew. Retrieved from http://www.sciencemag.org/news/2010/04/mystery-lionfish-dont-blame-hurricane-andrew

M. Roser, M. Dunbabin, & A. Geiger. (2014). Simultaneous underwater visibility assessment, enhancement and improved stereo. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3840–3847). https://doi.org/10.1109/ICRA.2014.6907416

National Geographic. (2016, August 29). Saving the Reef: Lionfish in Florida – National Geographic Blog. Retrieved from https://blog.nationalgeographic.org/2016/08/29/saving-the-reef-lionfish-in-florida/

Nave, Carl. “Electric Field.” *Hyperphysics*, http://hyperphysics.phy-astr.gsu.edu/hbase/electric/elefie.html. Accessed 13 Mar. 2019.

Neto, Iran E. Lima. "Maximum suction lift of water jet pumps." *Journal of Mechanical Science and Technology* 25.2 (2011): 391-394. from https://link.springer.com/content/pdf/10.1007%2Fs12206-010-1221-7.pdf

NOAA. (2018, June 25). Why are lionfish a threat to Atlantic Ocean fish? Retrieved October 8, 2018, from https://oceanservice.noaa.gov/facts/lionfish.html

PG series gearboxes. Retrieved from https://www.andymark.com/products/hex-pg-series-gearboxes-options

Platt, J. (2016, January 1). A Starfish-Killing, Artificially Intelligent Robot Is Set to Patrol the Great Barrier Reef. Retrieved October 8, 2018, from https://www.scientificamerican.com/article/a-starfish-killing-artificially-intelligent-robot-is-set-to-patrol-the-great-barrier-reef/

Polyurethane for marine applications. Retrieved from https://polyurethane.americanchemistry.com/Polyurethane-for-Marine-Applications/

Raj, B. (2018, April 11). Data Augmentation | How to use Deep Learning when you have Limited Data - Part 2. Retrieved from https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Raspberry Pi. (2018). Raspberry Pi 3 Model B. Retrieved December 13, 2018, from https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Reitermanov, Z. (2010). Data Splitting. In *WDS’10 Proceedings of Contributed Papers* (pp. 31–36). Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic: MATFYZPRESS. Retrieved from https://www.mff.cuni.cz/veda/konference/wds/proc/pdf10/WDS10_105_i1_Reitermanova.pdf

REEF. (2018). Lionfish Derbies | Reef Environmental Education Foundation. Retrieved October 8, 2018, from https://www.reef.org/lionfish-derbies

RSE. (2018). Lionfish Project. Retrieved October 8, 2018, from https://www.robotsise.org/lionfish-project/

Rössel, Johannes. (2012). Own work by uploader, based on File:Ejector or Injector.png, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=5558906

Schmoldt, A., Benthe, H. F., & Haberland, G. (1975). Digitoxin metabolism by rat liver microsomes. *Biochemical Pharmacology*, *24*(17), 1639–1641.

Skeeze. (2015, Jan 4). *Lionfish Tropical Venomous* [Photograph]. Retrieved from https://pixabay.com/en/lionfish-tropical-venomous-reef-587709/

Sparks, Sherry. “Electro-Fishing Summary.” *Petitcodiac - Environmental Impact Assessment*, 3 Apr. 2007, https://web.archive.org/web/20070403201413/http://www.petitcodiac.com/eFishing/electroFishing.htm.

Szegedy, C., Vanhoucke, V., Loffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. Retrieved December 14, 2018, from https://arxiv.org/abs/1512.00567.

TensorFlow. (2018). TensorFlow Lite. Retrieved December 13, 2018, from https://www.tensorflow.org/lite/

TensorFlow Lite | TensorFlow. (2018). Retrieved October 5, 2018, from https://www.tensorflow.org/lite/

Tchou, Patrick, and Mark Kroll. “How a Taser Works.” *IEEE Spectrum: Technology, Engineering, and Science News*, 30 Nov. 2007, https://spectrum.ieee.org/consumer-electronics/gadgets/how-a-taser-works.

US Department of Commerce, & National Oceanic and Atmospheric Administration. (2004, December 19). NOAA National Ocean Service Education: Lionfish Discovery Story. Retrieved from https://oceanservice.noaa.gov/education/stories/lionfish/lion02_invade.html

US Department of Commerce, & National Oceanic and Atmospheric Administration. (2016, July 28). What is a lionfish? Retrieved from https://oceanservice.noaa.gov/facts/lionfish-facts.html

Venturi, Giovanni Battista. *Experimental enquiries concerning the principle of the lateral communication of motion in fluids [electronic resource]: Applied to the explanation of various hydraulic phenomena. By citizen J.B. Venturi, prosessor of Natural Philosophy at Modena, member of the Italian Society, of the Institute of Bologna, of the Agrarian Society of Turin, &c. Translated from the French* Printed for J. Taylor, at the Architectural Library, High-Holborn London 1799

Waterproofing a DC motor with sugru. (2011). Retrieved from http://www.robotshop.com/community/forum/t/waterproofing-a-dc-motor-with-sugru/13077

Winoto, S. H., H. Li, and D. A. Shah. "Efficiency of jet pumps." *Journal of Hydraulic Engineering* 126.2 (2000): 150-156. Retrieved from https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9429%282000%29126%3A2%28150%29

White, Frank Mangrom., and Rhim Yoon. Chul. *Fluid Mechanics*. 8th ed., McGraw-Hill Education, 2016.

WHOOSHH INNOVATIONS - What We Do. (n.d.). Retrieved from https://www.whooshh.com/what-we-do.html

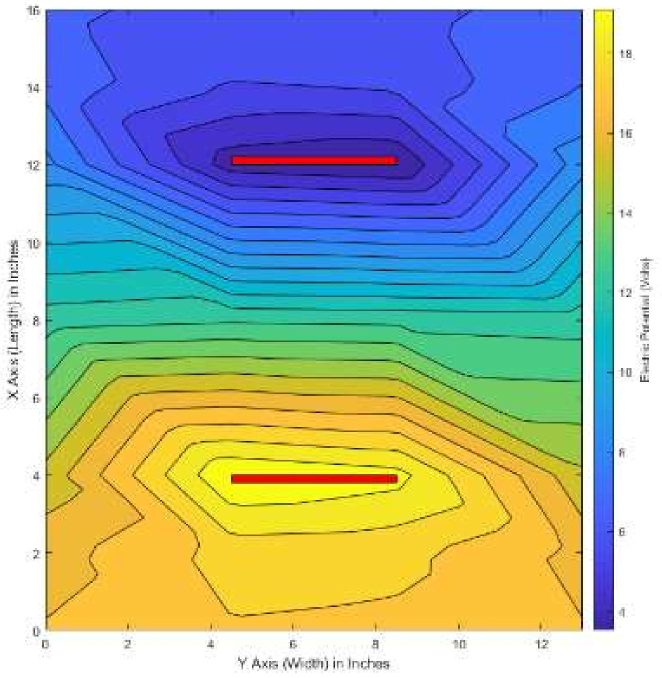
Wood, R. J. K. (2006). Erosion–corrosion interactions and their effect on marine and offshore materials. *Wear, 261*(9), 1012-1023. doi:10.1016/j.wear.2006.03.033

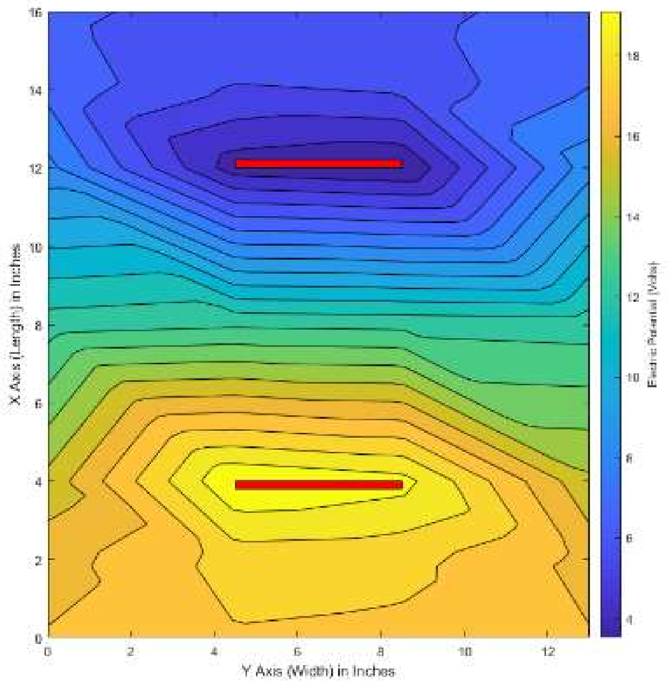
(n.d.). Retrieved from http://cs231n.github.io/convolutional-networks/

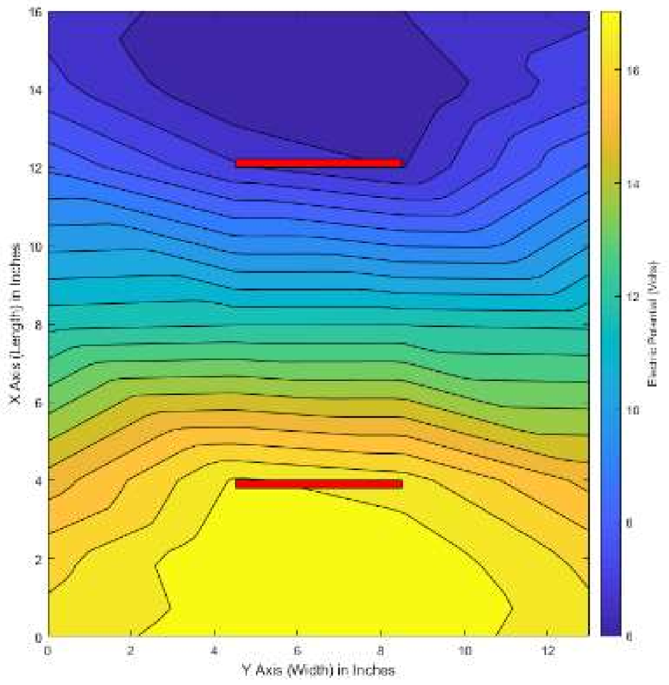
Appendix A

A.1 Panel Voltage Maps

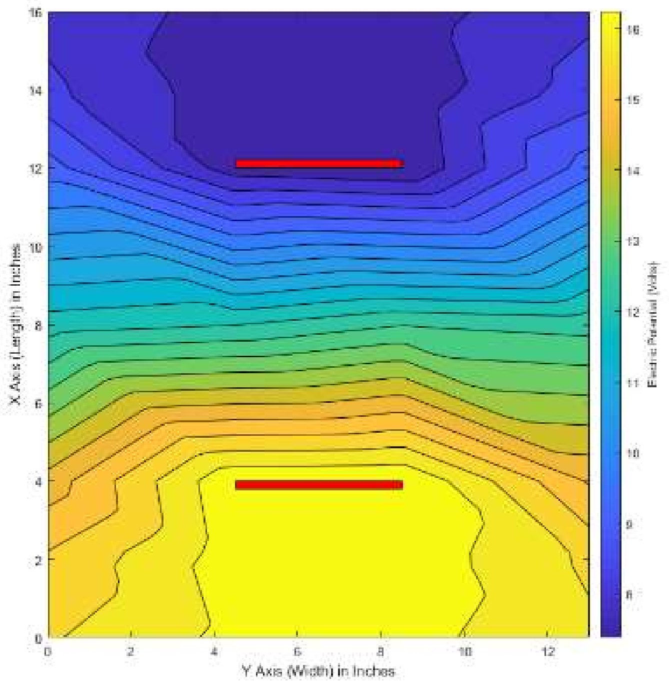
In Section 3.1.8, we discussed the mapping of the voltage potential throughout the basin for each of our panel shapes. Appendix A provides a comprehensive collection of all the voltage maps we generated from our results.



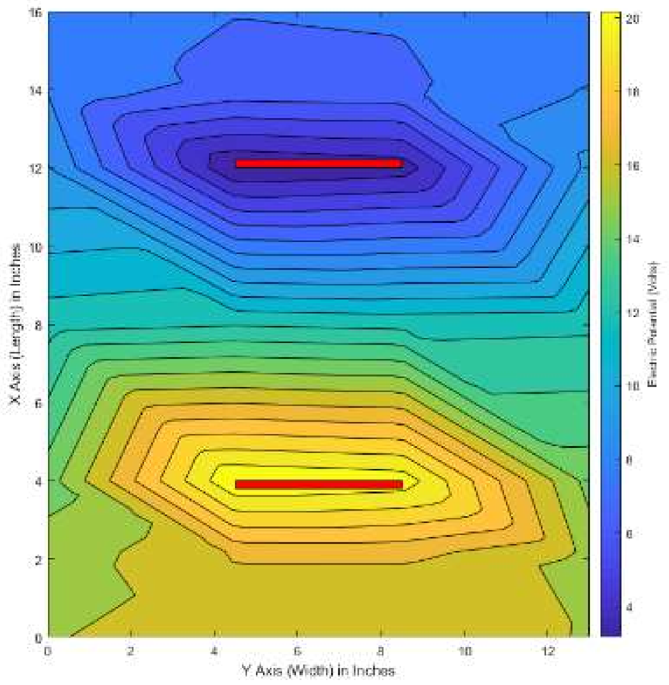




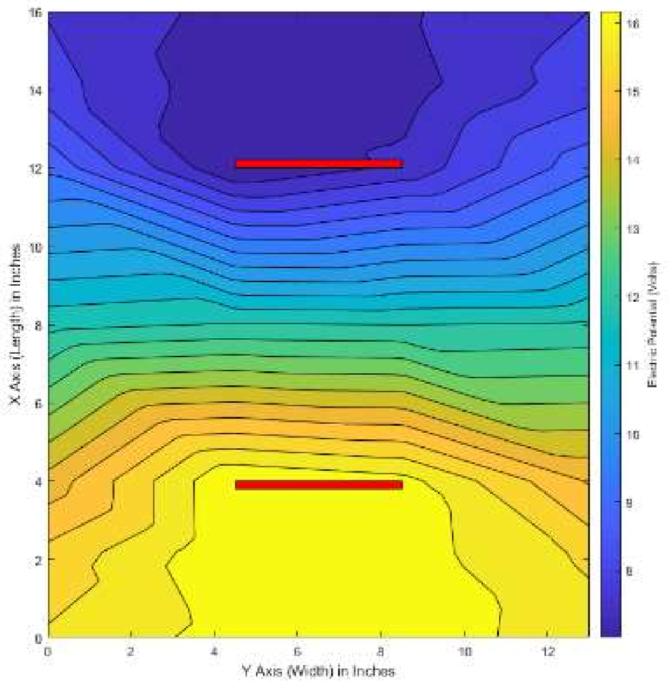
Large Square Panel – Top-Down – Potential Maps



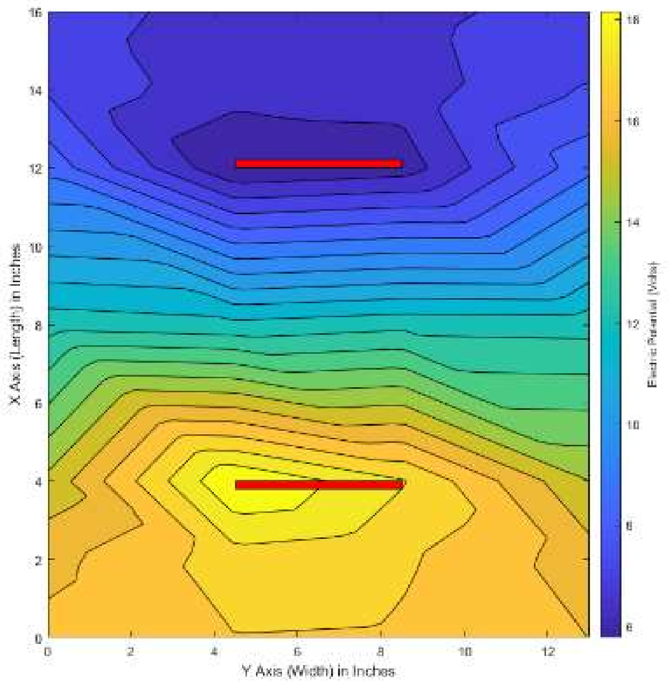
X Axis (Length) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)

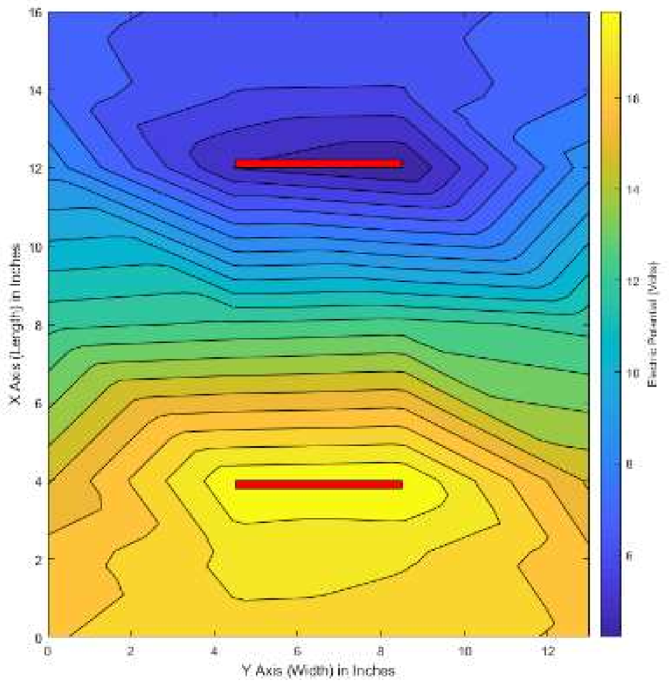


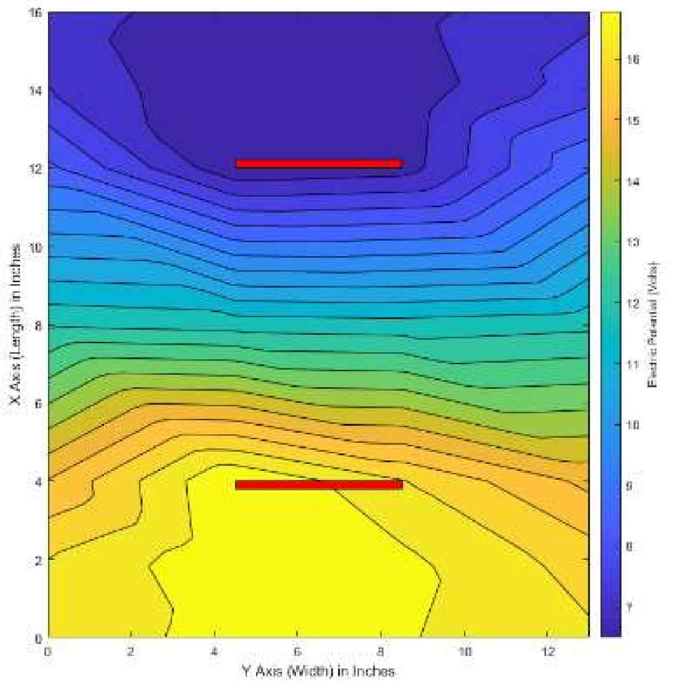
X Axis (Length) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)



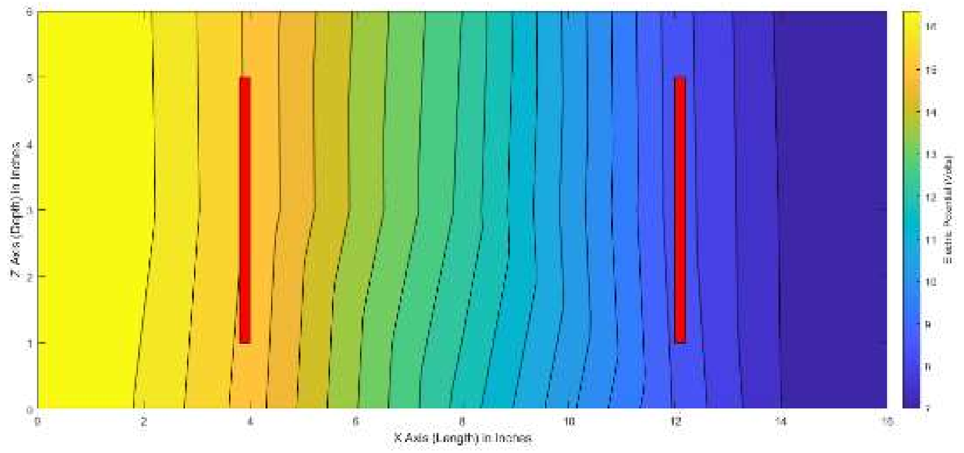
Circle Panel – Top-Down – Potential Maps

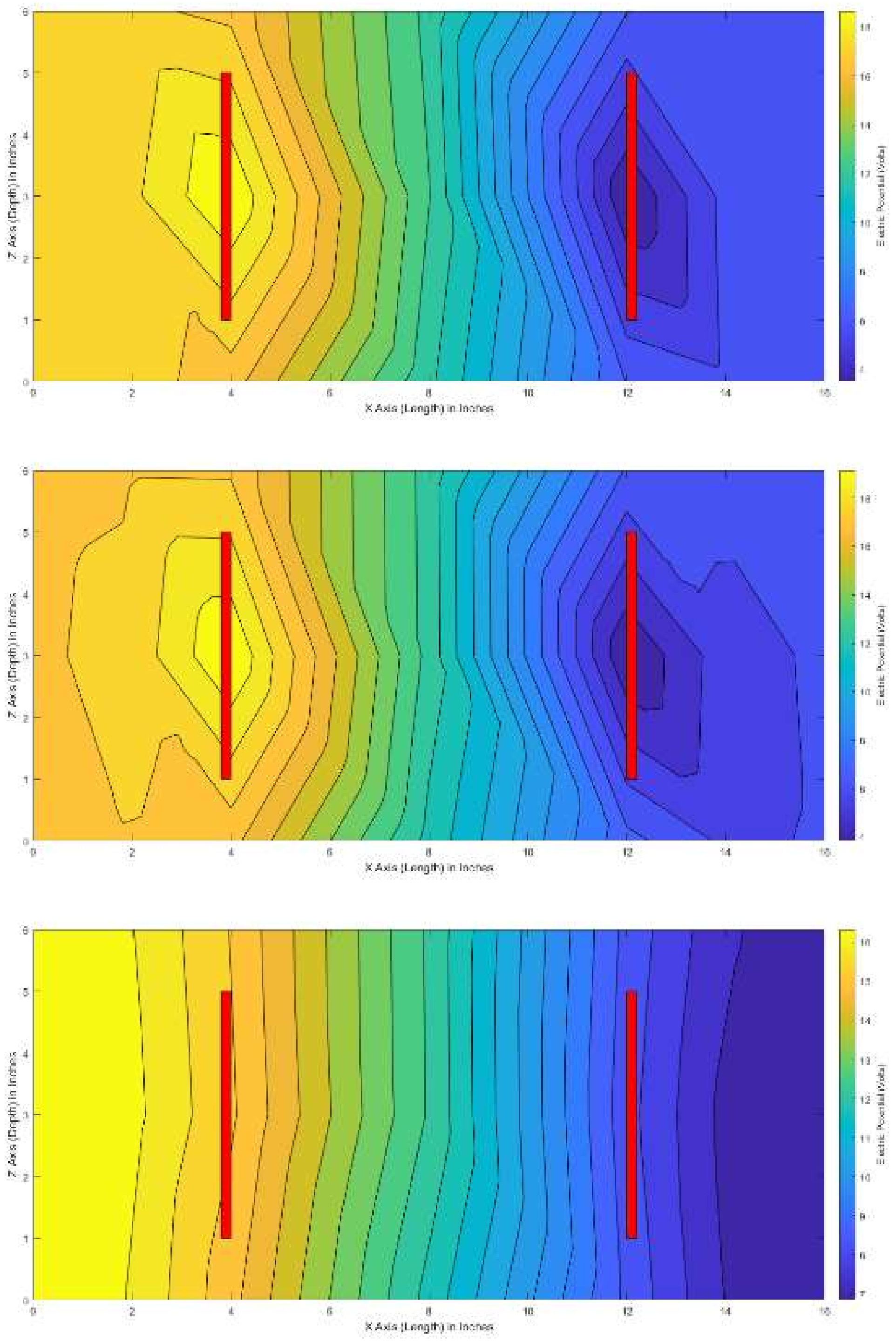

X Axis (Length) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)


X Axis (Length) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)

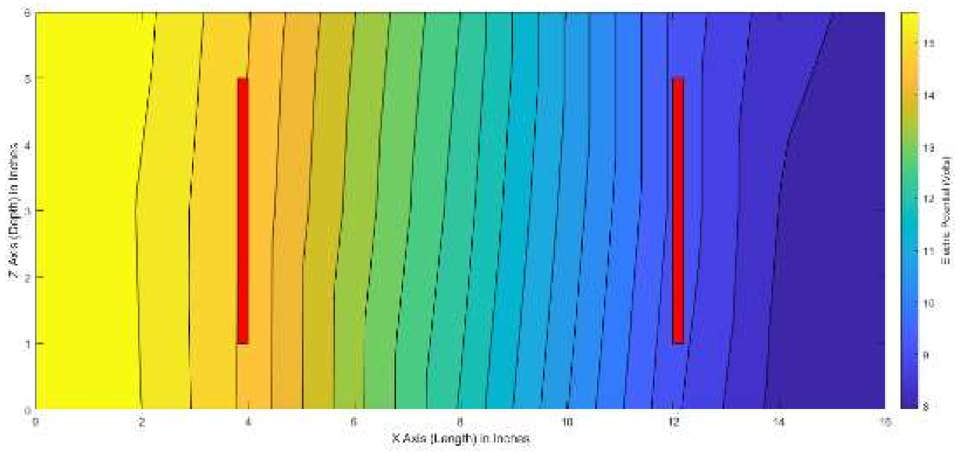


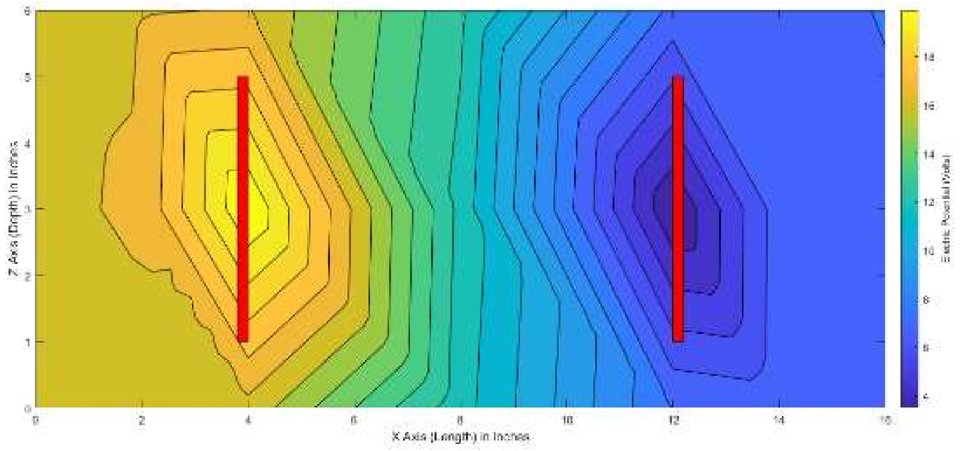
Cross Panel – Top-Down – Potential Maps

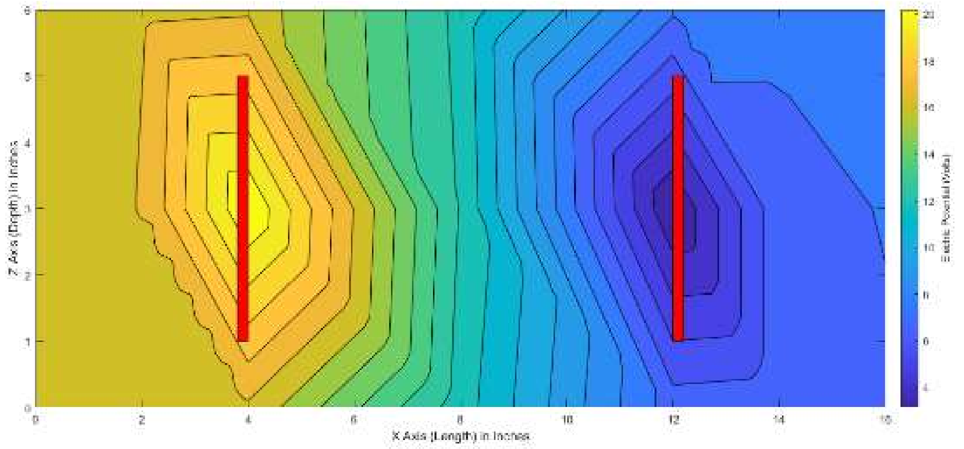


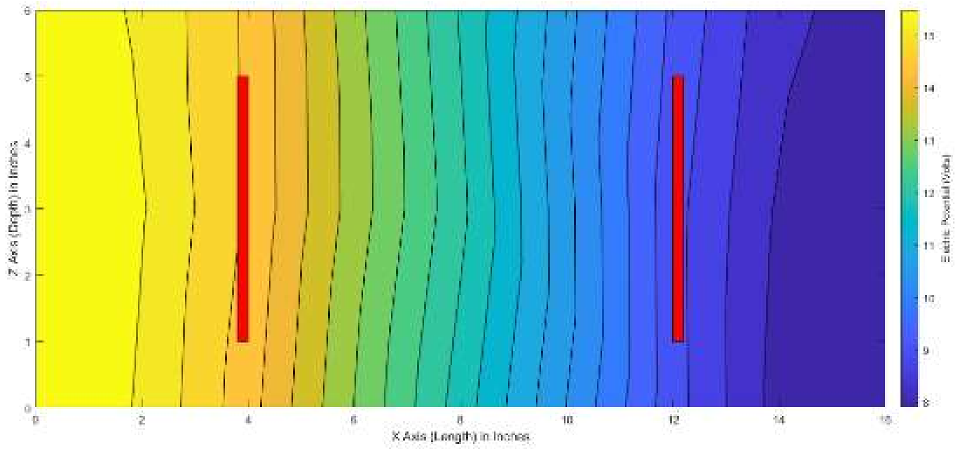


Large Square Panel – Vertical – Potential Maps

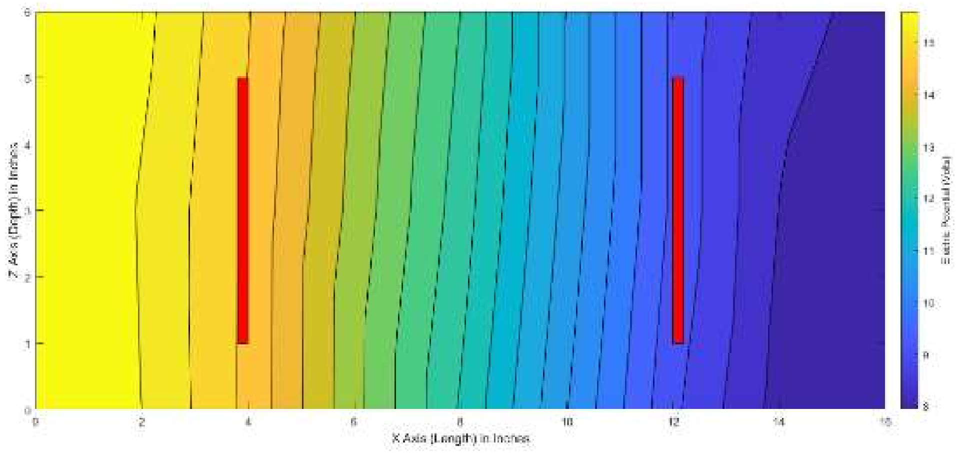

Z Axis (Depth) in Inches
X Axis (Length) in Inches
Electric Potential (Volts)

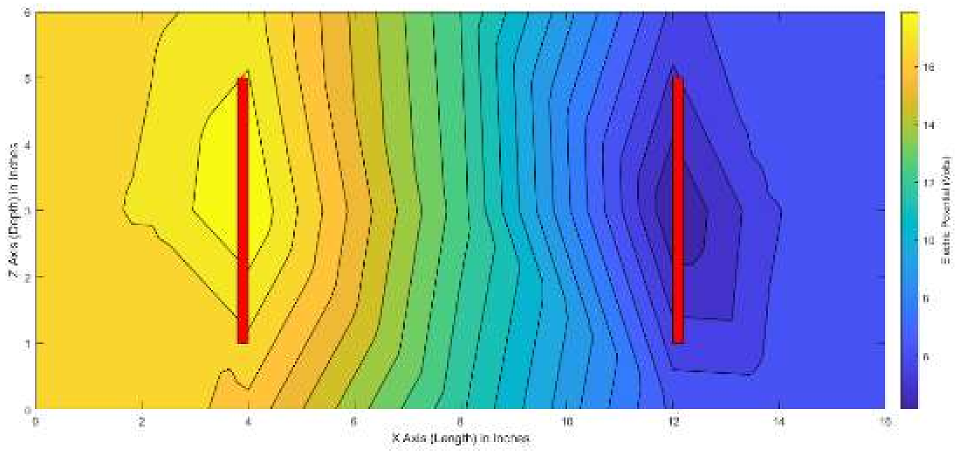

Z Axis (Depth) in Inches
X Axis (Length) in Inches
Electric Potential (Volts)

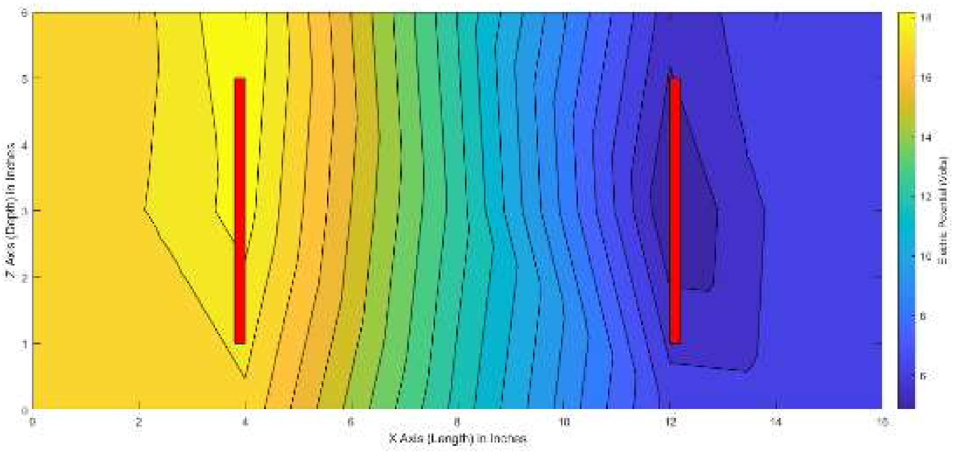

Z Axis (Depth) in Inches
X Axis (Length) in Inches
Electric Potential (Volts)

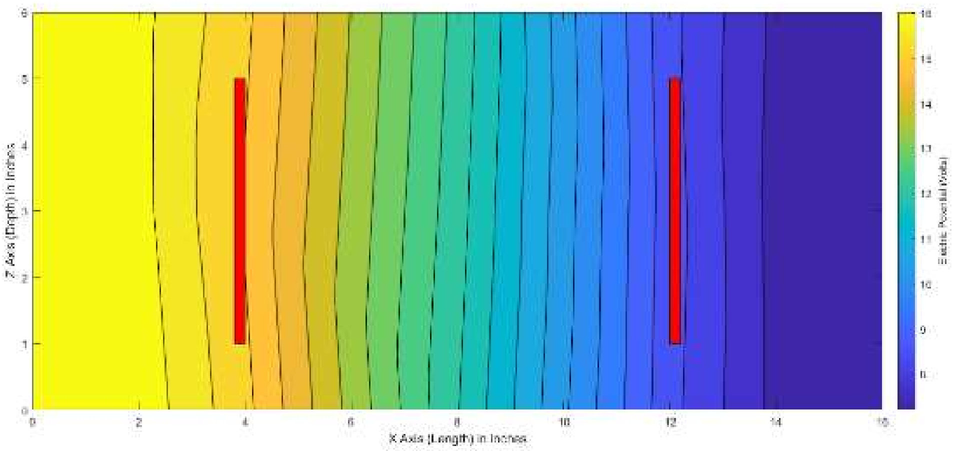


Circle Panel – Vertical – Potential Maps

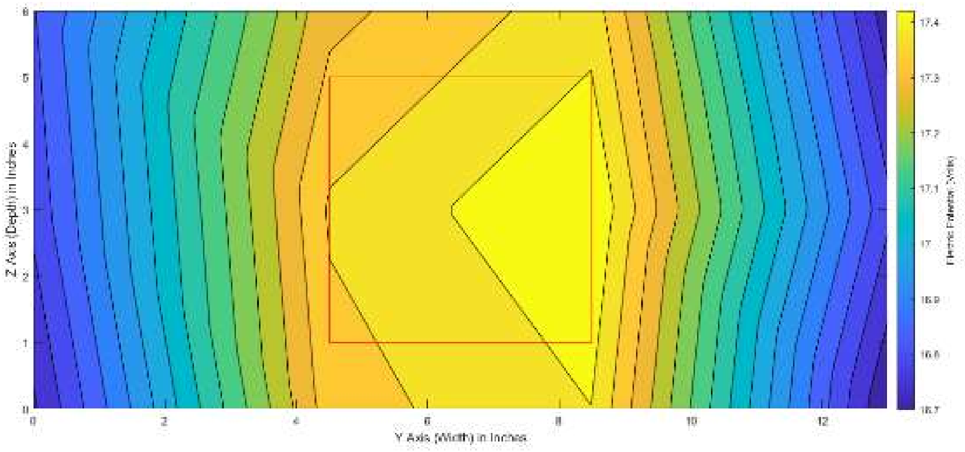


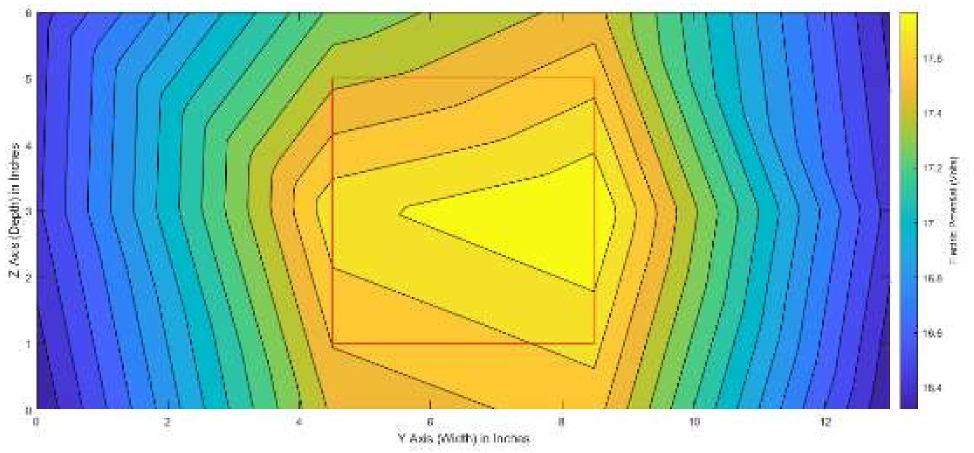


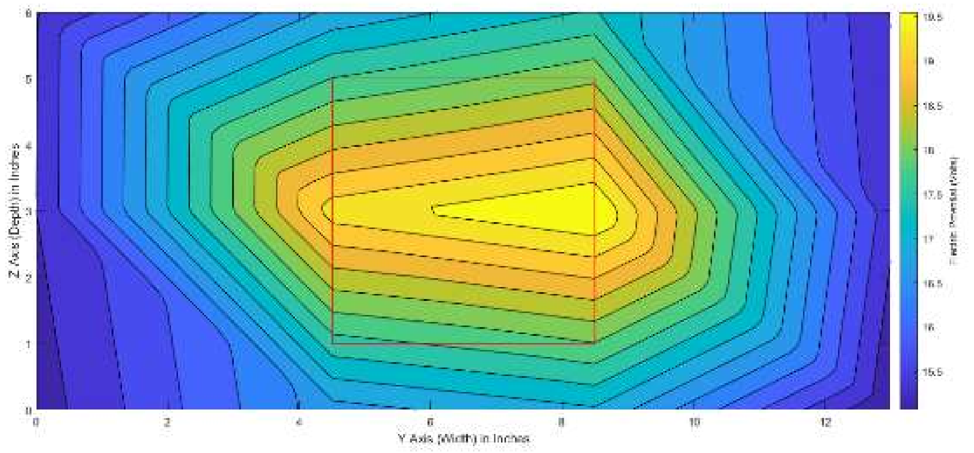


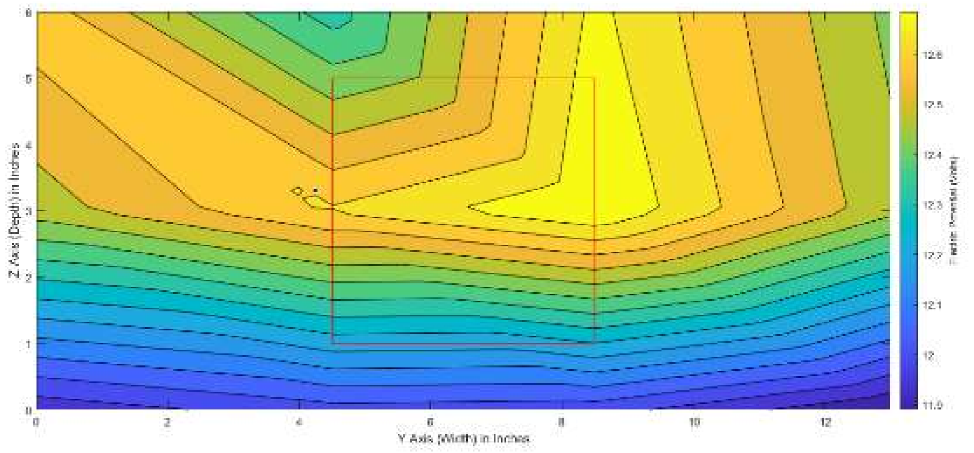


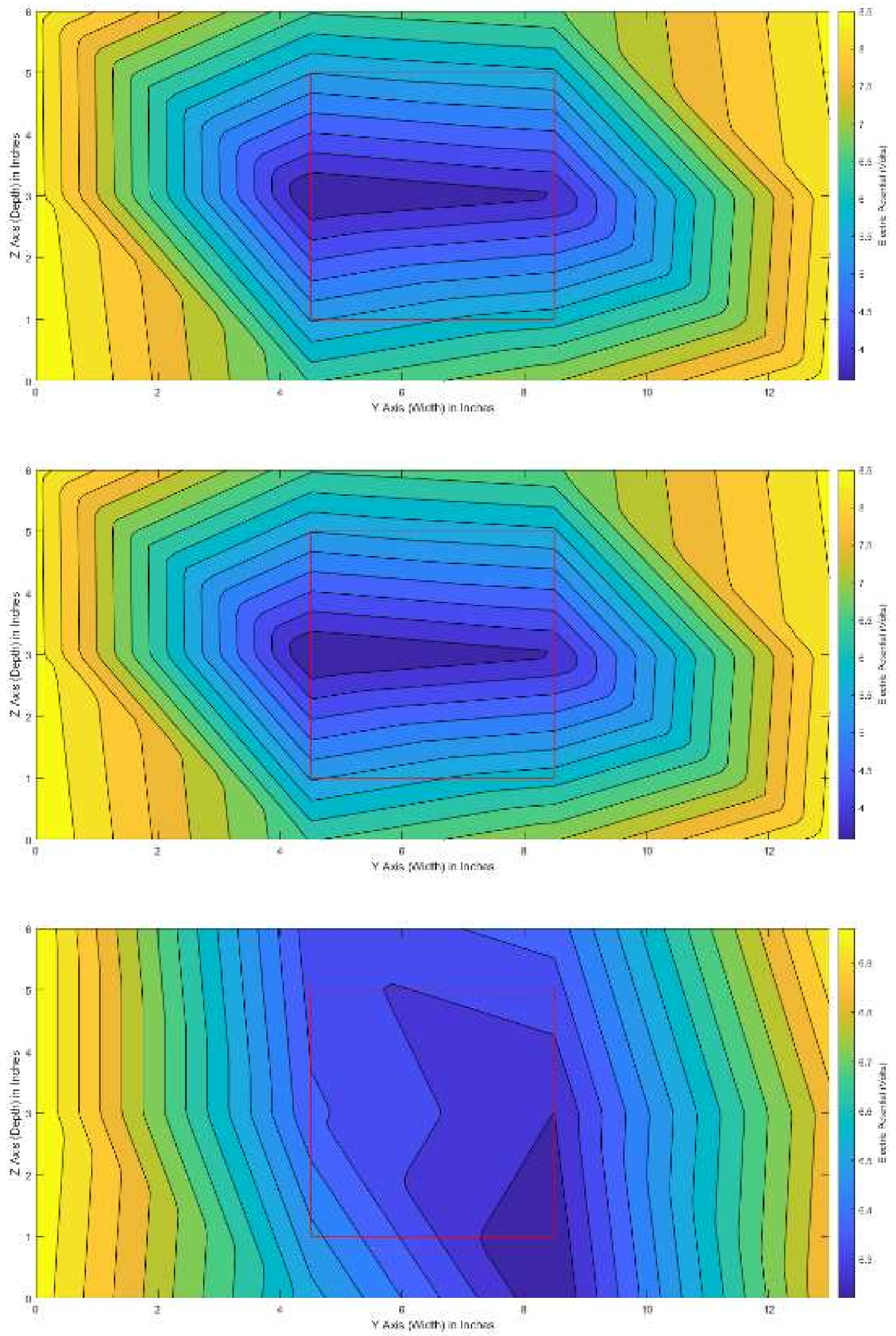
Cross Panel – Vertical – Potential Maps



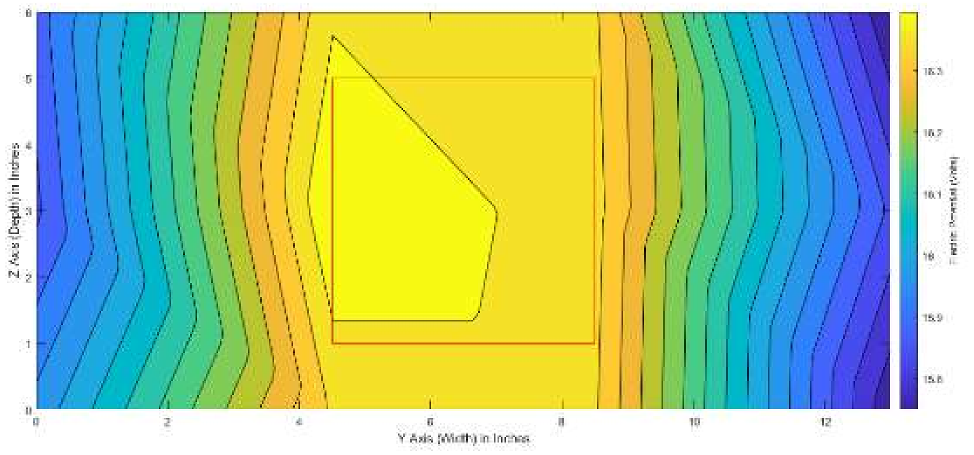

Z Axis (Depth) in Inches
0
1
2
3
4
5
6
Y Axis (Width) in Inches
0
2
4
6
8
10
12
Electric Potential (Volts)
16.4
16.6
16.8
17
17.2
17.4
17.6

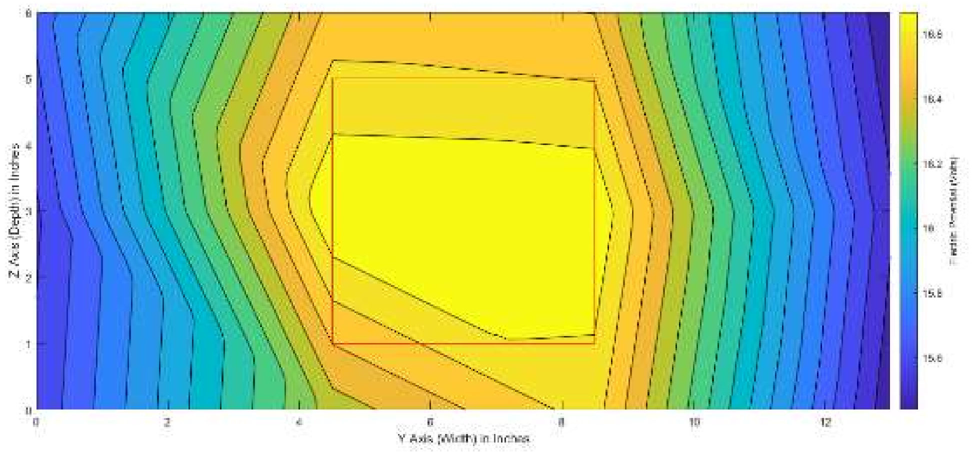

Z Axis (Depth) in Inches
0
1
2
3
4
5
6
Y Axis (Width) in Inches
0
2
4
6
8
10
12
Electric Potential (Volts)
15.5
16
16.5
17
17.5
18
18.5
19
19.5

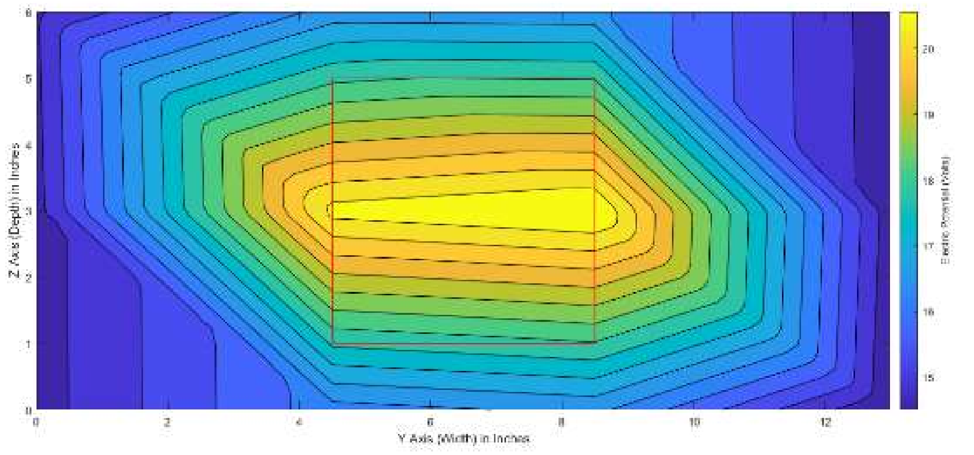

Z Axis (Depth) in Inches
0
1
2
3
4
5
6
Y Axis (Width) in Inches
0
2
4
6
8
10
12
Electric Potential (Volts)
11.9
12
12.1
12.2
12.3
12.4
12.5
12.6

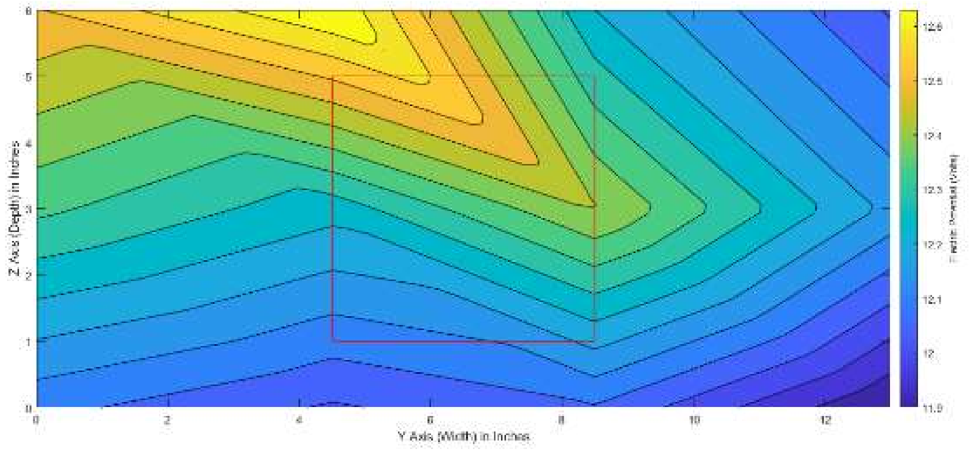


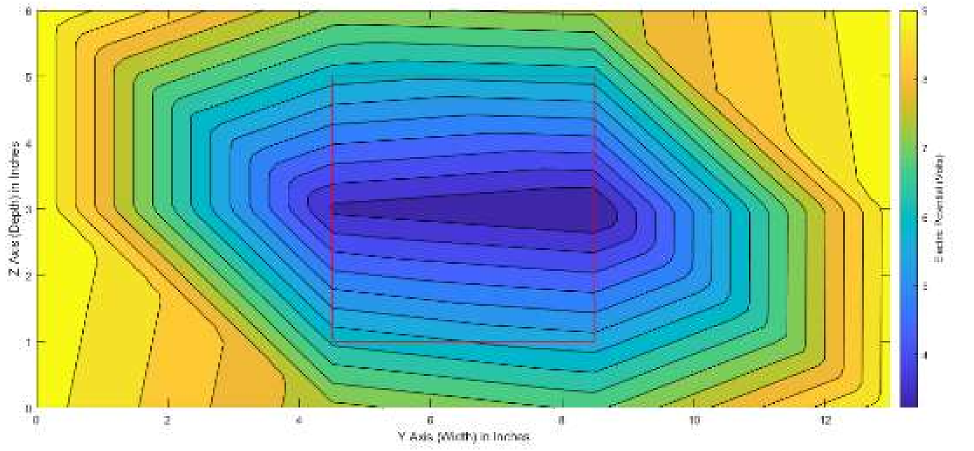
Large Square Panel – Cross-Cut – Potential Map

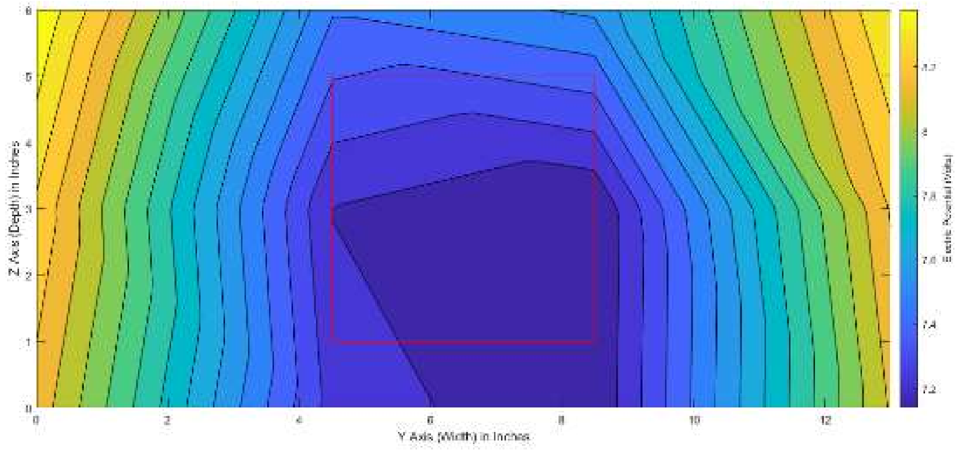

Z Axis (Depth) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)
16.3
16.2
16.1
16
15.9
15.8

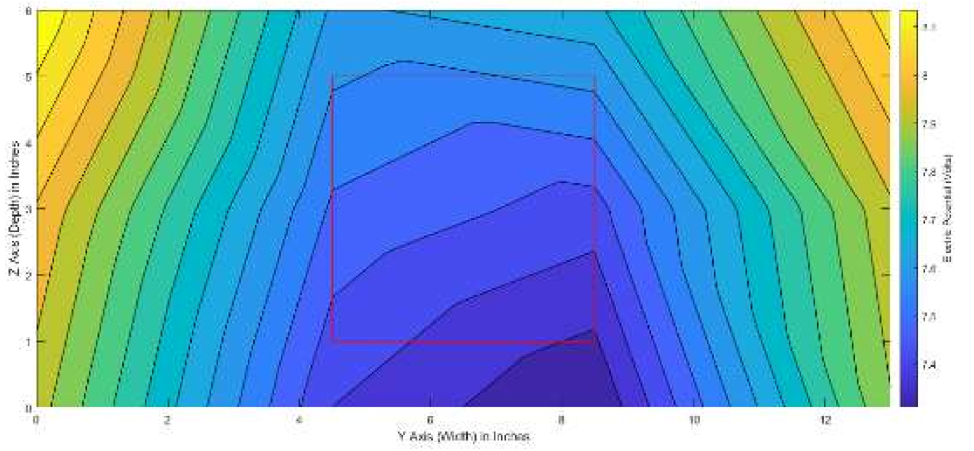

Z Axis (Depth) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)
16.6
16.4
16.2
16
15.8
15.6


Z Axis (Depth) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)
20
19
18
17
16
15

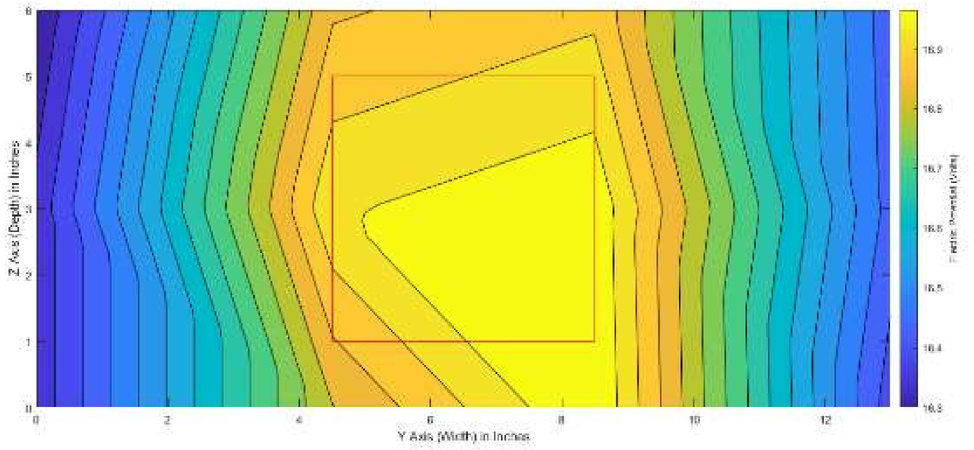

Z Axis (Depth) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)

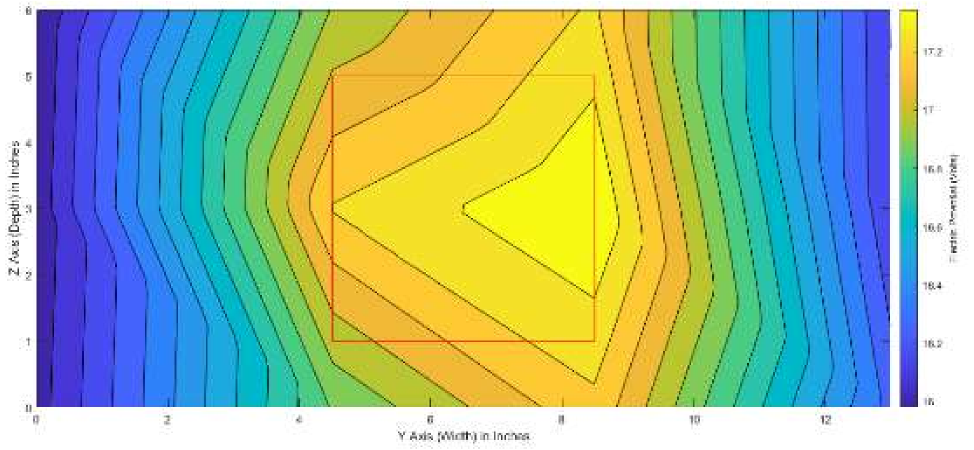

Z Axis (Depth) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)

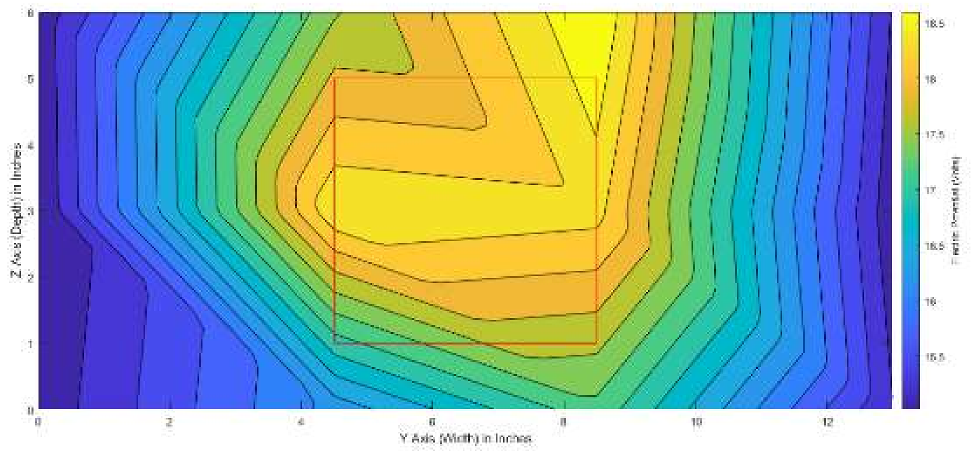

Z Axis (Depth) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)

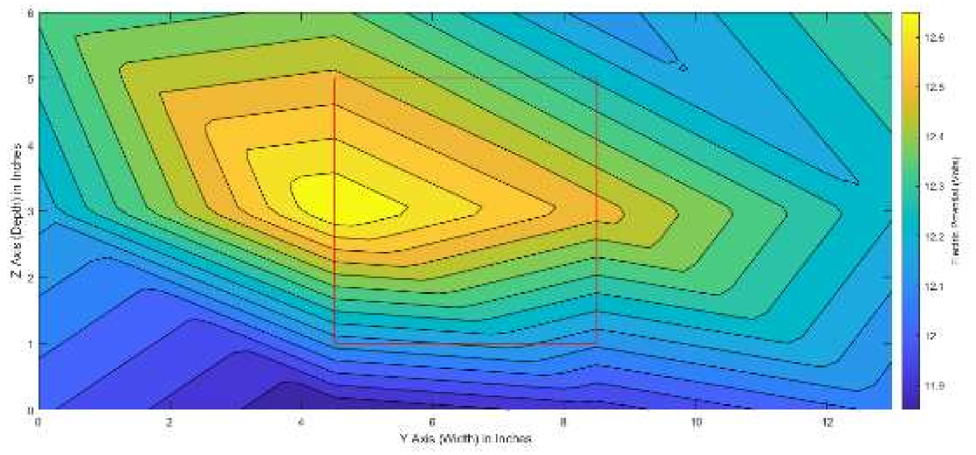


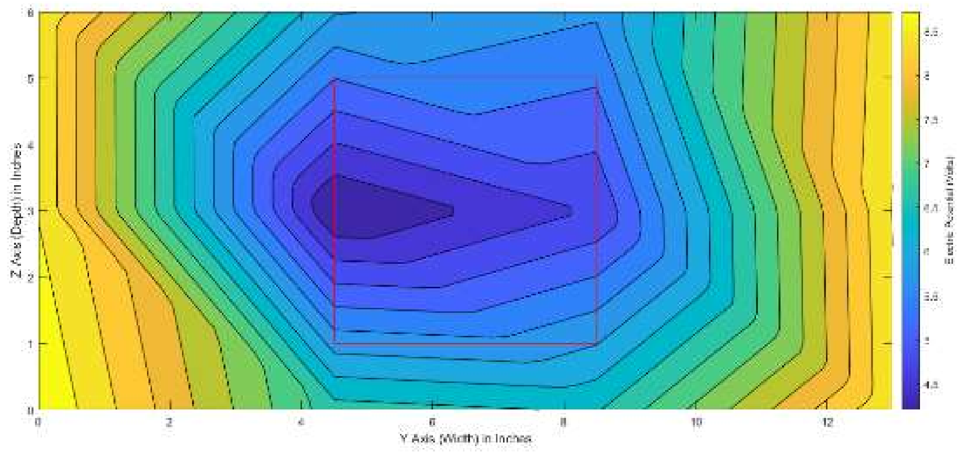
Circle Panel – Vertical – Potential Maps

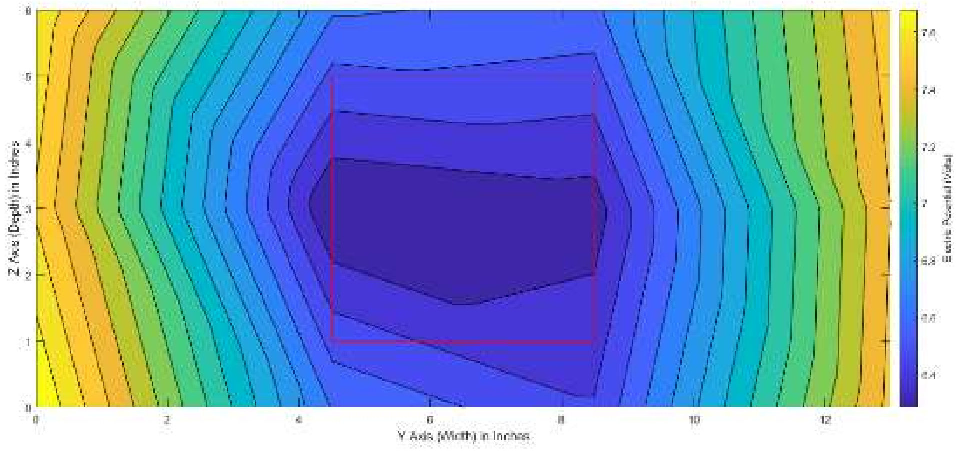


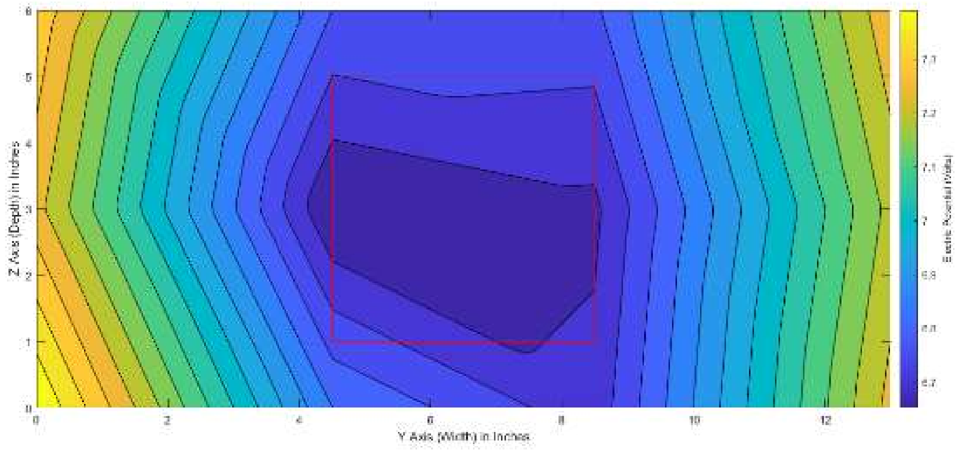



Z Axis (Depth) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)
15.5
16
16.5
17
17.5
18
18.5


Z Axis (Depth) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)
11.9
12
12.1
12.2
12.3
12.4
12.5
12.6


Z Axis (Depth) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)
4.5
5
5.5
6
6.5
7
7.5
8
8.5

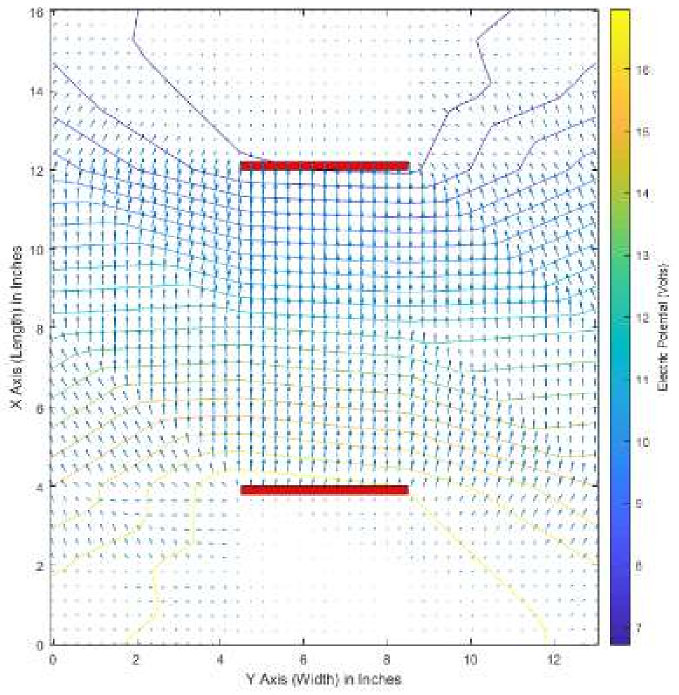


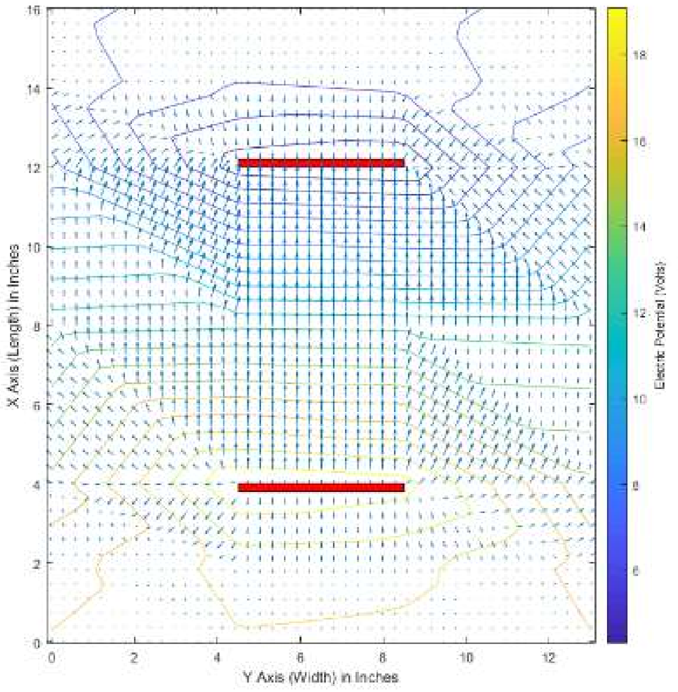


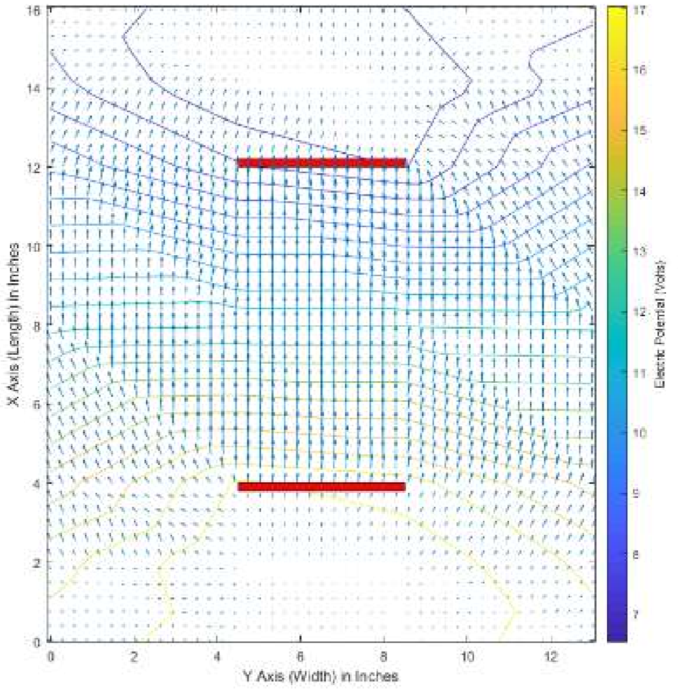
Cross Panel – Vertical – Potential Maps

A.2 Field Plots

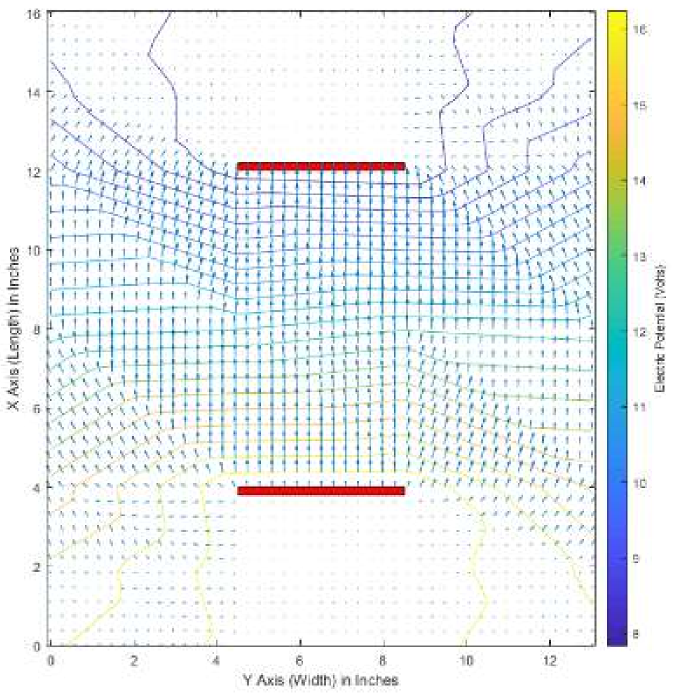
In Section 3.1.9, we discussed the field maps that were generated from the voltage potential maps. Appendix A.2 provides a comprehensive collection of the field maps we generated.

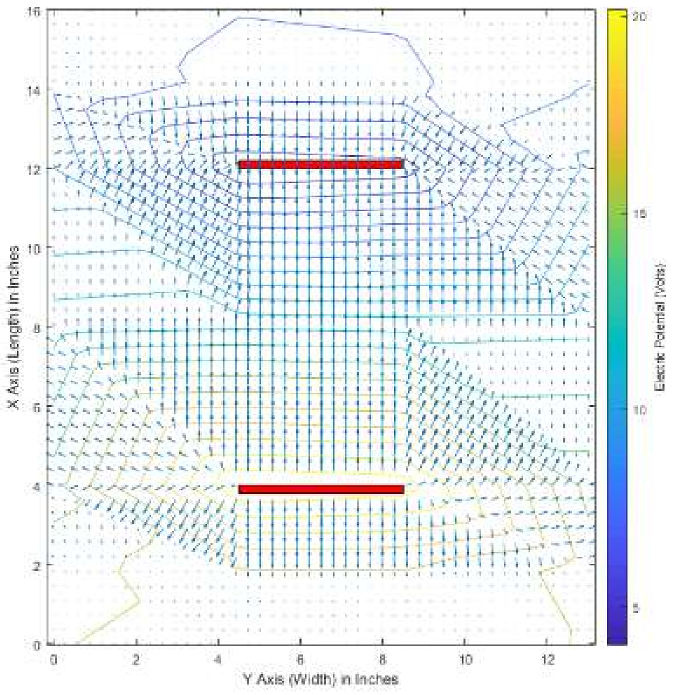


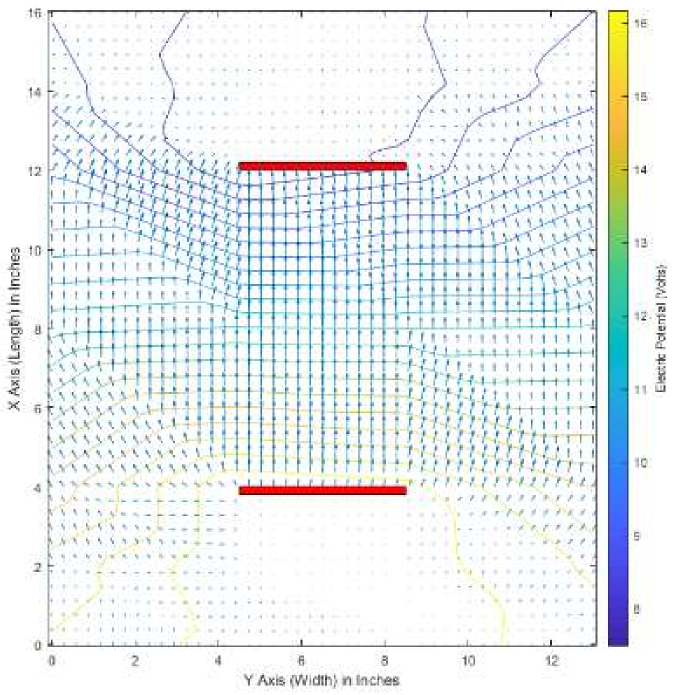




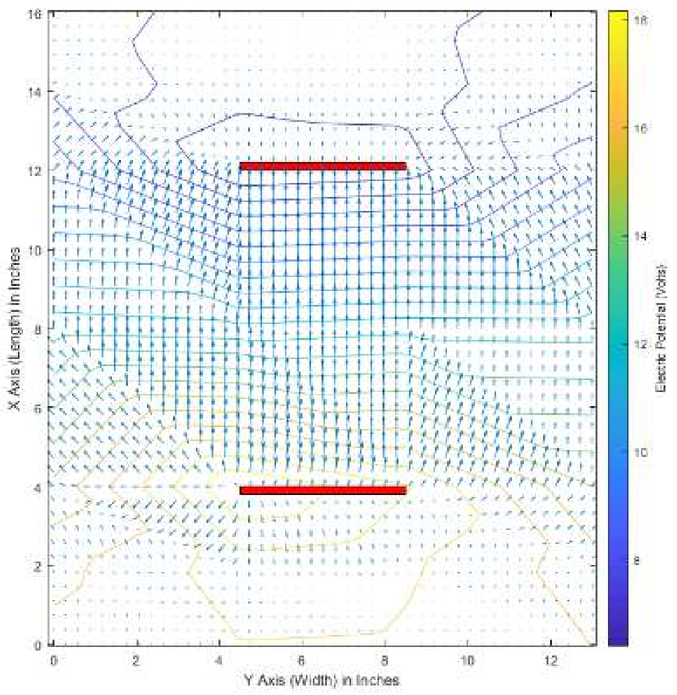
Large Square Panel – Top-Down – Field Maps

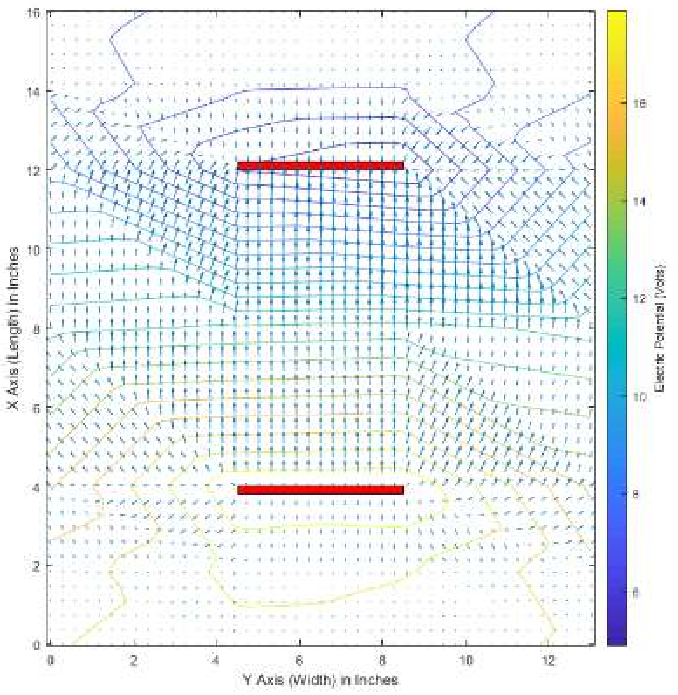

X Axis (Length) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)

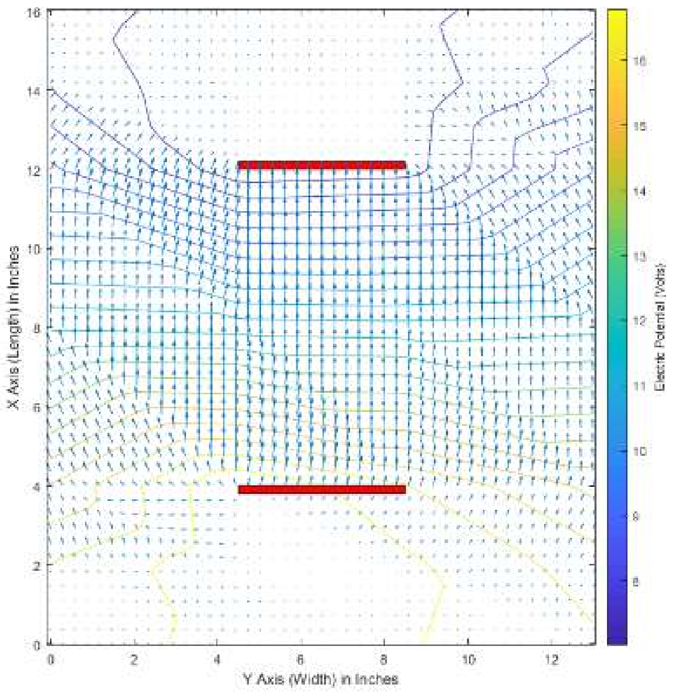

X Axis (Length) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)



Circle Panel – Top-Down – Field Maps



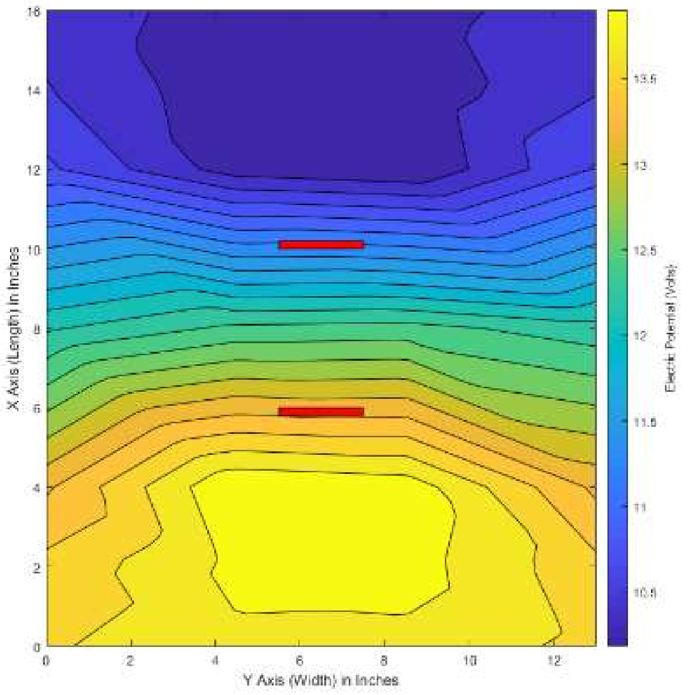


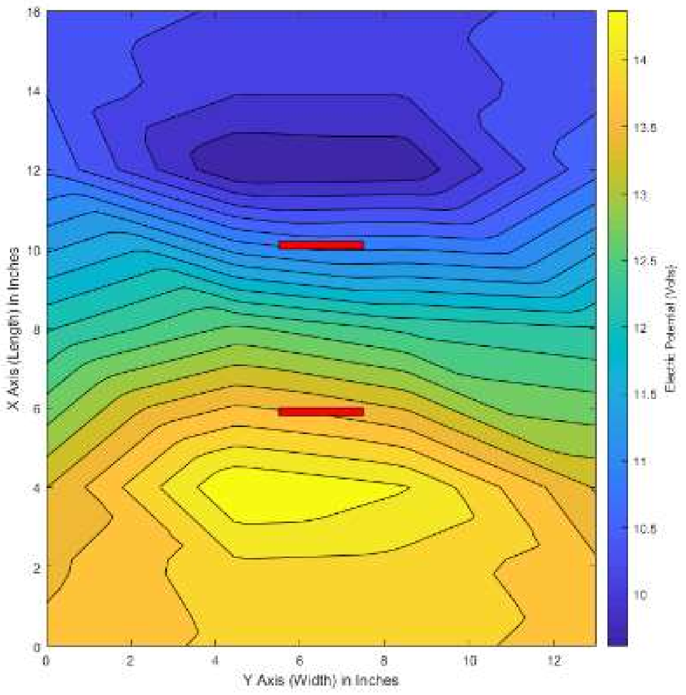


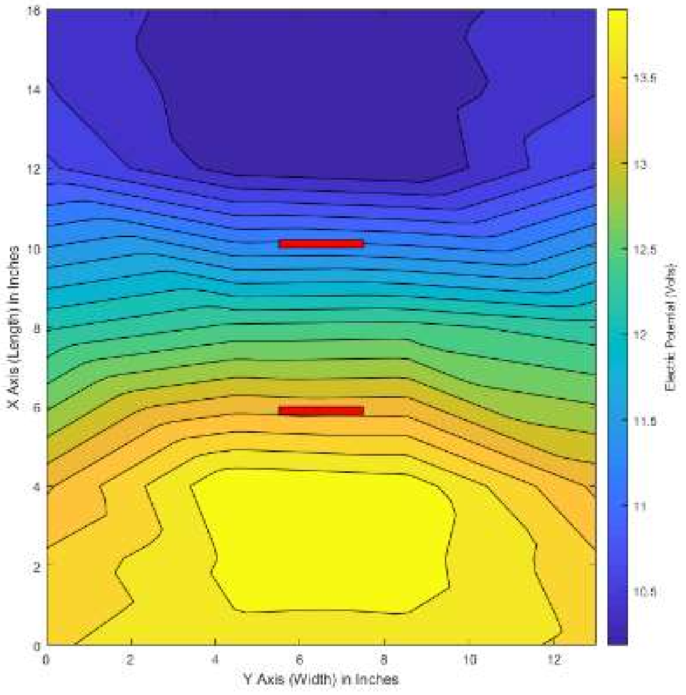
Cross Panel – Top-Down- Field Maps

A.3 Small Panel Test Results

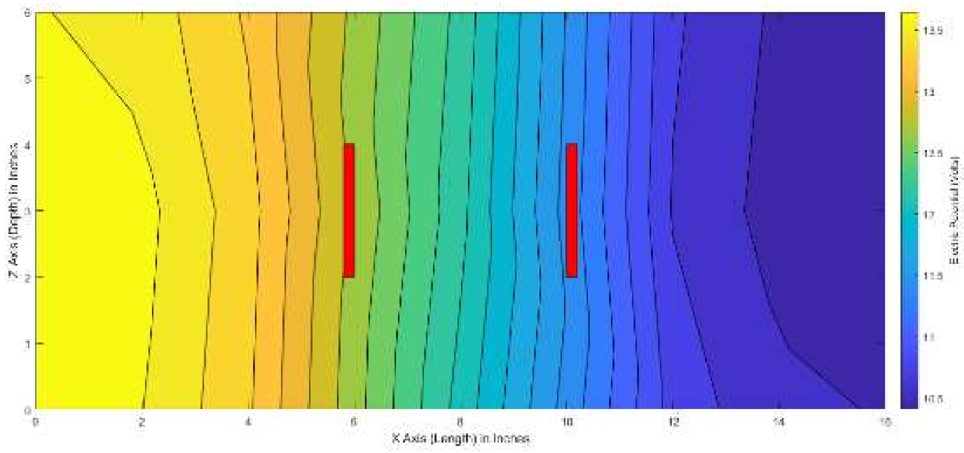
In Section 3.1 we discussed the use of a smaller set of panels designed to allow us to see the field out to a farther relative distance. This was used to determine the properties of the field near the edge of the basin. These results have been compiled here in Appendix A.3.

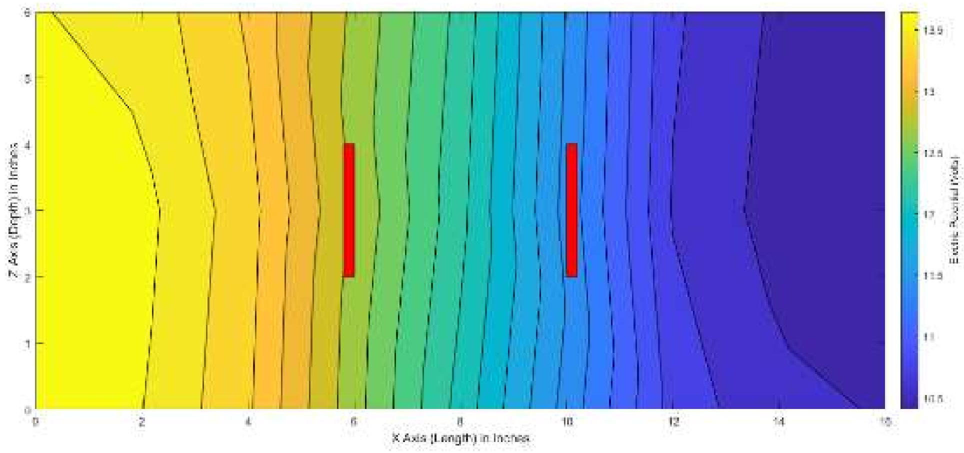


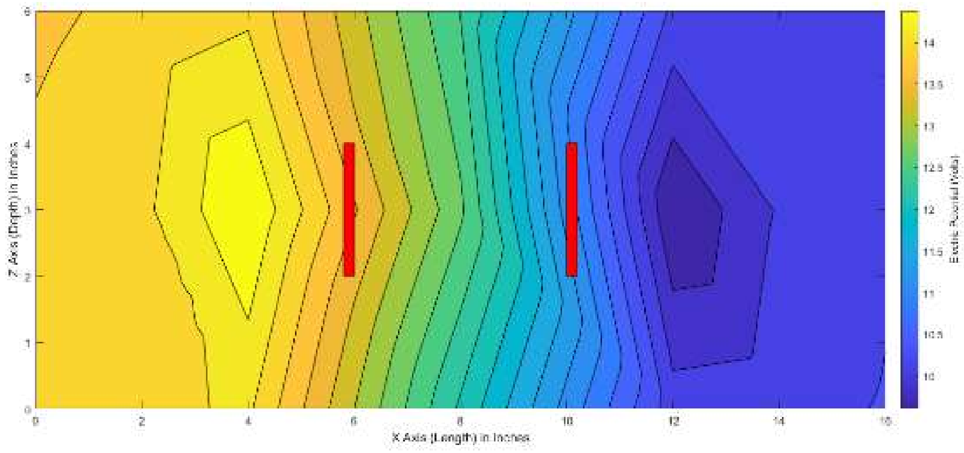


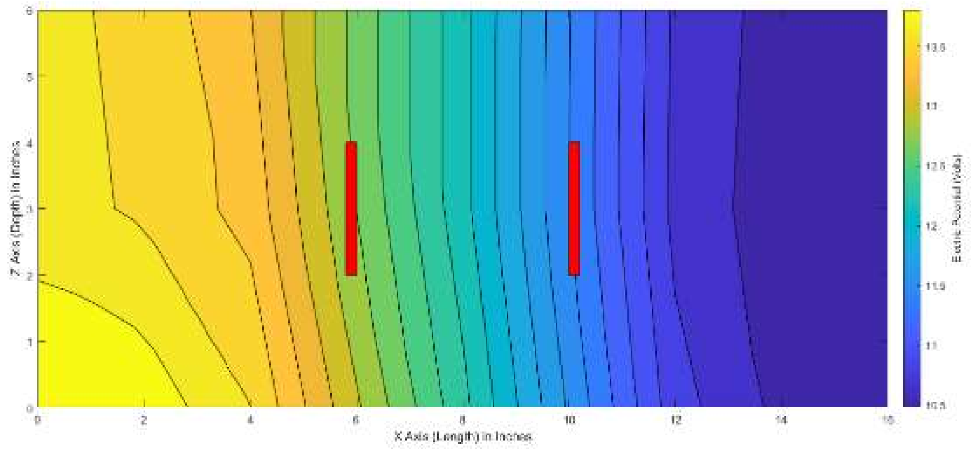


Small Panel – Top-Down – Potential Maps

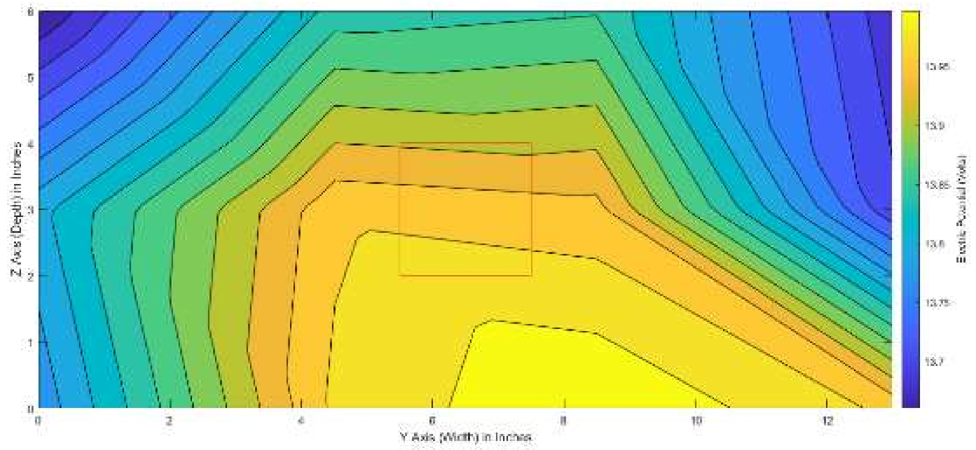

Z Axis (Depth) in Inches
X Axis (Length) in Inches
Electric Potential (Volts)
13.5
13
12.5
12
11.5
11
10.5

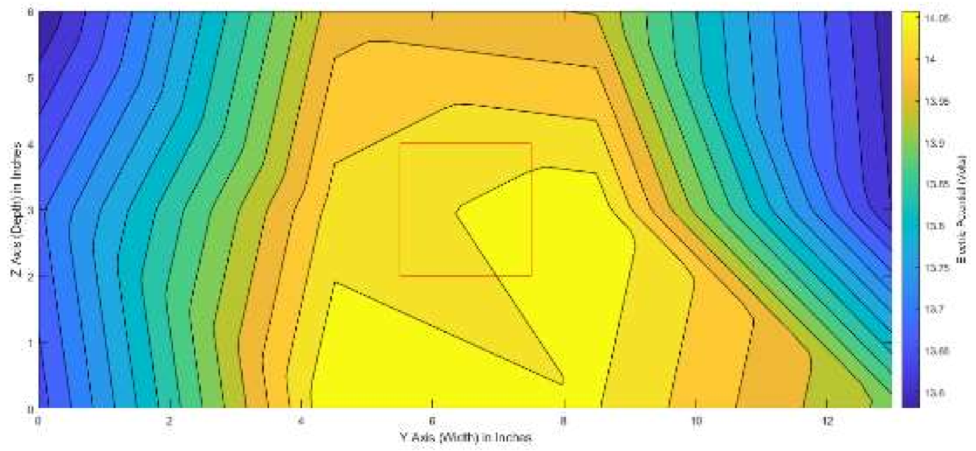

Z Axis (Depth) in Inches
X Axis (Length) in Inches
Electric Potential (Volts)
13.5
13
12.5
12
11.5
11
10.5

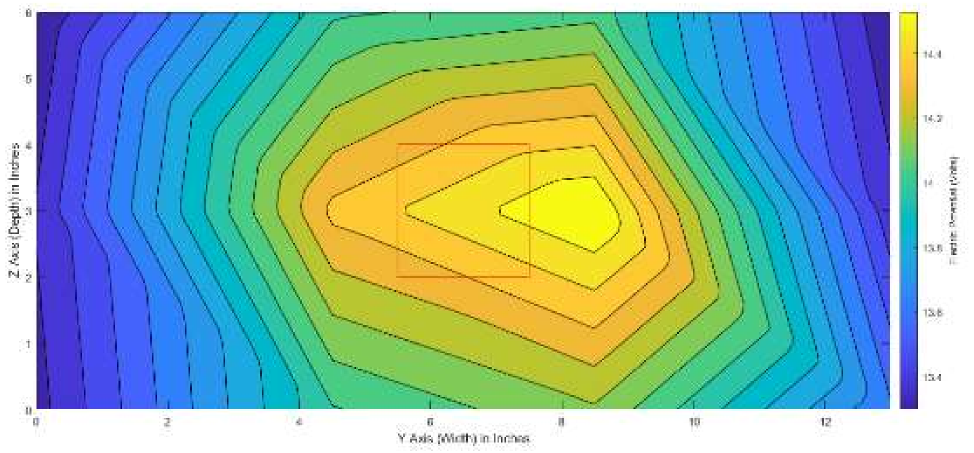

Z Axis (Depth) in Inches
X Axis (Length) in Inches
Electric Potential (Volts)
14
13.5
13
12.5
12
11.5
11
10.5
10

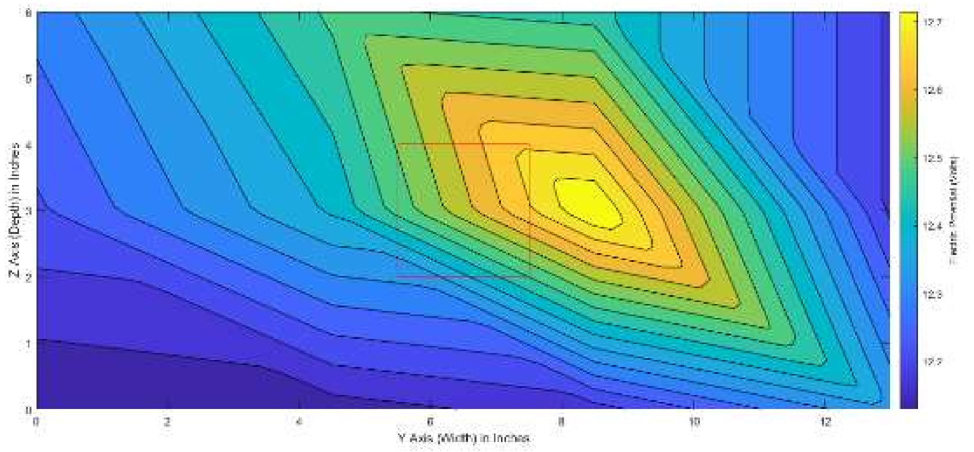


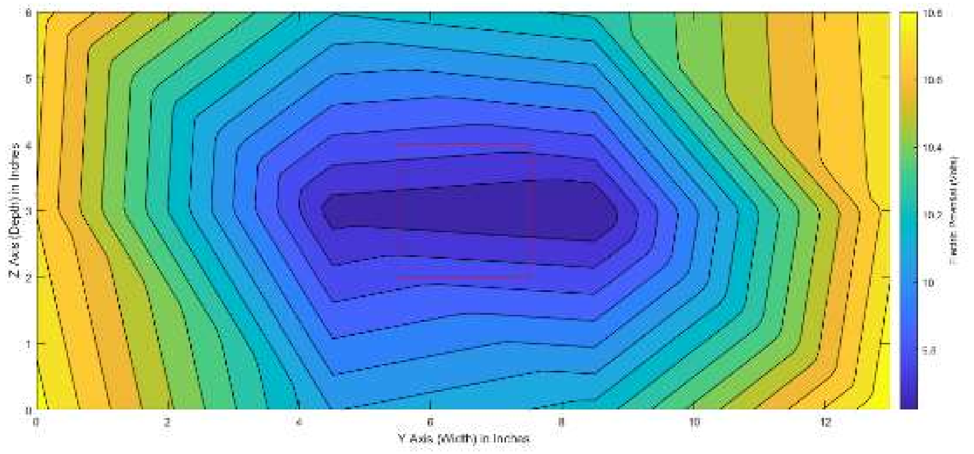
Small Panel – Vertical – Potential Maps

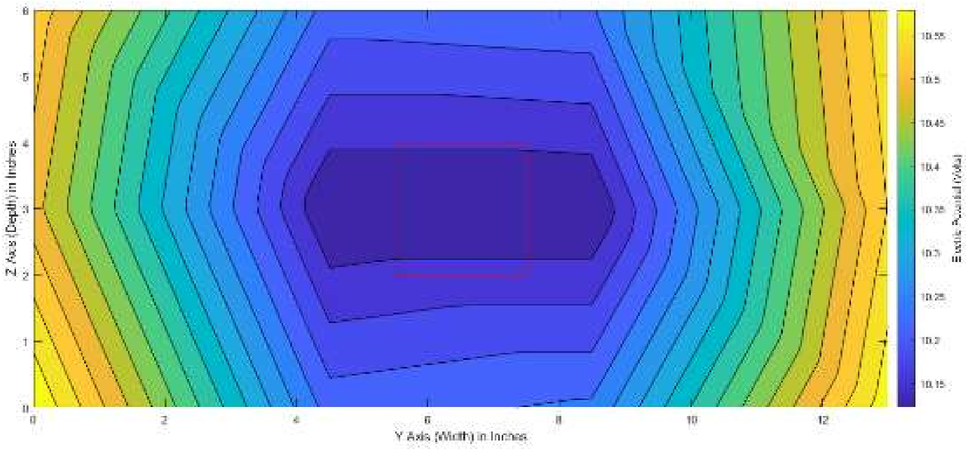


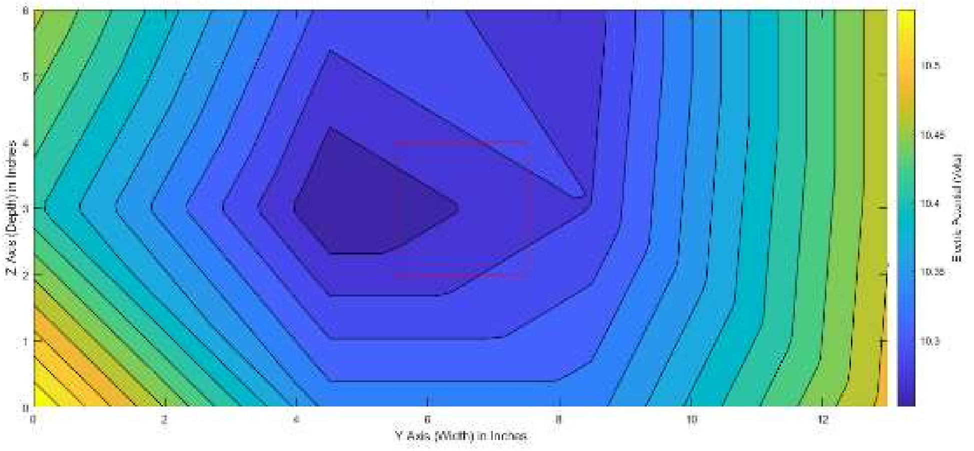



Z Axis (Depth) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)

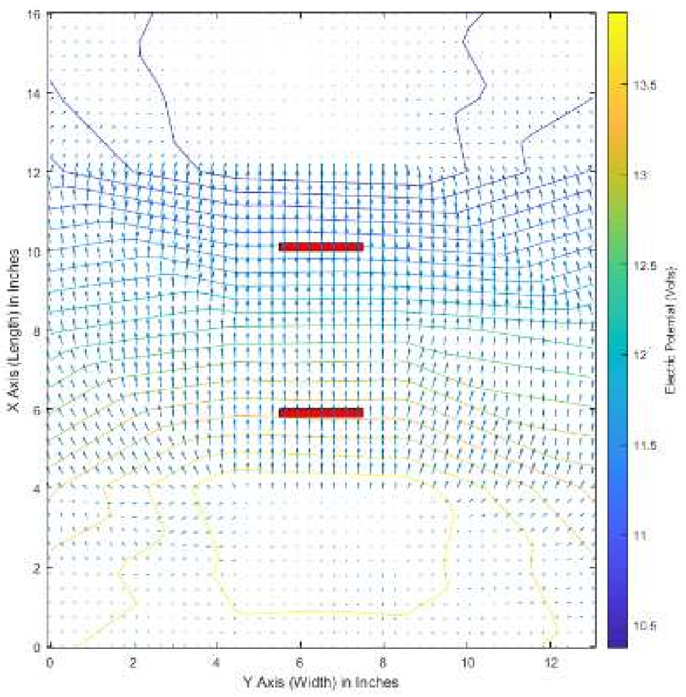

Z Axis (Depth) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)


Z Axis (Depth) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)

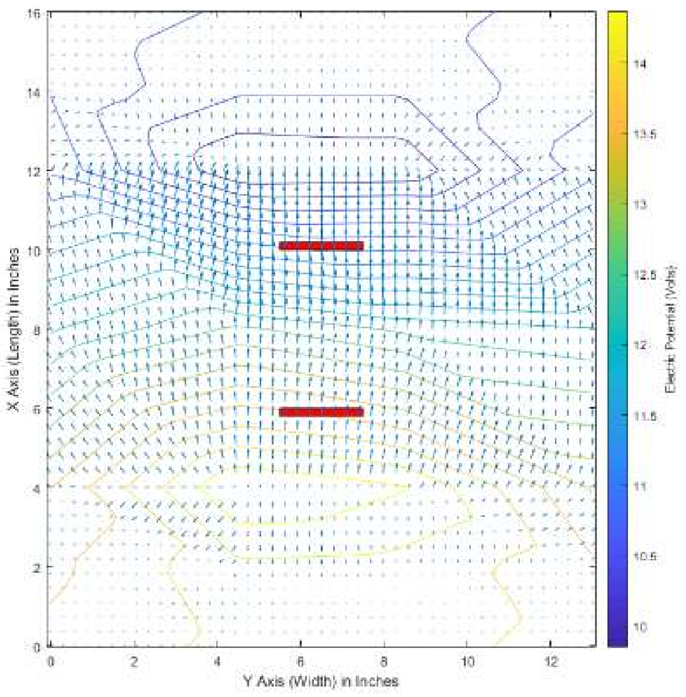




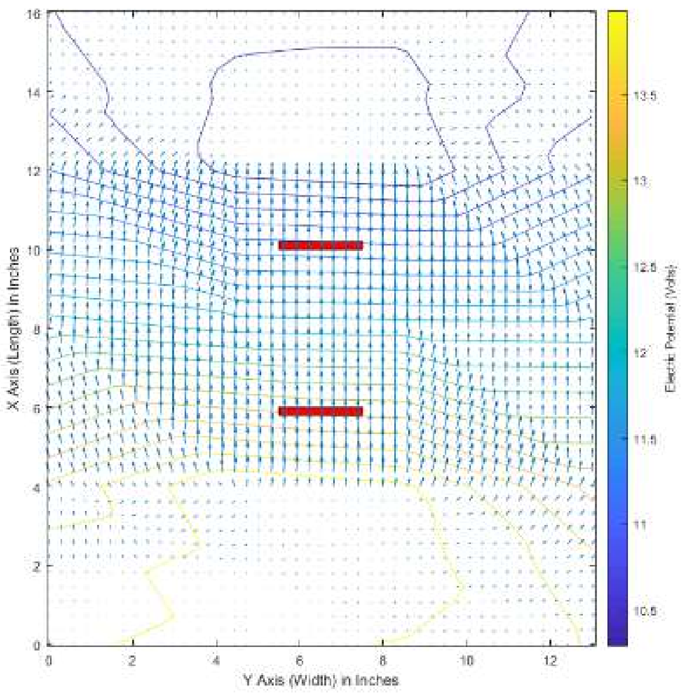
Small Panel – Cross-Cut – Potential Maps



X Axis (Length) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)



X Axis (Length) in Inches
Y Axis (Width) in Inches
Electric Potential (Volts)



Small Panel – Top-Down – Field Maps

A.4 Panel Data Points

In Section 3.1.8 we discussed the use of data points in order to generate voltage potential maps. Appendix A.4 is a comprehensive list of all the numerical voltage values we recorded at each location.

X	Y	Z		Voltage
0	0	0		16.74
0	4.5	0		17.35
0	8.5	0		17.42
0	13	0		16.7
2	0	0		16.33
2	4.5	0		17.5
2	8.5	0		17.62
2	13	0		16.32
4	0	0		15.21
4	4.5	0		16.74
4	8.5	0		17.12
4	13	0		15.07
8	0	0		11.96
8	4.5	0		12.03
8	8.5	0		12.02
8	13	0		11.89
12	0	0		8.83
12	4.5	0		6.5
12	8.5	0		7.16
12	13	0		8.64
14	0	0		7.43
14	4.5	0		5.96
14	8.5	0		5.98
14	13	0		7.27
16	0	0		7
16	4.5	0		6.54
16	8.5	0		6.22
16	13	0		6.87
0	0	3		16.81
0	4.5	3		17.38
0	8.5	3		17.47
0	13	3		16.81
2	0	3		16.49
2	4.5	3		17.74
2	8.5	3		17.87
2	13	3		16.47
4	0	3		15.37
4	4.5	3		19.42

X	Y	Z		Voltage
4	8.5	3		19.9
4	13	3		15.47
8	0	3		12.5
8	4.5	3		12.64
8	8.5	3		12.74
8	13	3		12.49
12	0	3		8.66
12	4.5	3		3.51
12	8.5	3		3.85
12	13	3		8.44
14	0	3		7.41
14	4.5	3		6.04
14	8.5	3		5.85
14	13	3		7.17
16	0	3		7.02
16	4.5	3		6.38
16	8.5	3		6.27
16	13	3		6.85
0	0	6		16.84
0	4.5	6		17.31
0	8.5	6		17.4
0	13	6		16.72
2	0	6		16.46
2	4.5	6		17.31
2	8.5	6		17.52
2	13	6		16.38
4	0	6		15.35
4	4.5	6		16.94
4	8.5	6		17.49
4	13	6		15.36
8	0	6		12.61
8	4.5	6		12.34
8	8.5	6		12.7
8	13	6		12.46
12	0	6		8.62
12	4.5	6		6.57
12	8.5	6		6.8
12	13	6		8.59
14	0	6		7.37
14	4.5	6		6.15
14	8.5	6		6.35
14	13	6		7.4
16	0	6		7.02
16	4.5	6		6.34
16	8.5	6		6.39
16	13	6		6.97

Square Panel Data Points

X	Y	Z		Voltage
0	0	0		16.36
0	4.5	0		16.83
0	8.5	0		17.01
0	13	0		16.41
2	0	0		16.03
2	4.5	0		16.9
2	8.5	0		17.23
2	13	0		16.31
4	0	0		15.1
4	4.5	0		16.3
4	8.5	0		17.33
4	13	0		15.27
8	0	0		12.02
8	4.5	0		11.85
8	8.5	0		11.93
8	13	0		12.08
12	0	0		9.02
12	4.5	0		6.74
12	8.5	0		6.58
12	13	0		8.67
14	0	0		7.77
14	4.5	0		6.65
14	8.5	0		6.48
14	13	0		7.52
16	0	0		7.44
16	4.5	0		6.85
16	8.5	0		6.72
16	13	0		7.26
0	0	3		16.36
0	4.5	3		16.96
0	8.5	3		17
0	13	3		16.46
2	0	3		16.01
2	4.5	3		17.26
2	8.5	3		17.44
2	13	3		16.17
4	0	3		15.02
4	4.5	3		18.58
4	8.5	3		18.46
4	13	3		15.08
8	0	3		12.19
8	4.5	3		12.71
8	8.5	3		12.52
8	13	3		12.23
12	0	3		8.7
12	4.5	3		4.17
12	8.5	3		4.84
12	13	3		8.73

X	Y	Z		Voltage
14	0	3		7.57
14	4.5	3		6.28
14	8.5	3		6.33
14	13	3		7.49
16	0	3		7.26
16	4.5	3		6.65
16	8.5	3		6.69
16	13	3		7.21
0	0	6		16.3
0	4.5	6		16.87
0	8.5	6		16.91
0	13	6		16.45
2	0	6		15.98
2	4.5	6		16.99
2	8.5	6		17.27
2	13	6		16.16
4	0	6		15.06
4	4.5	6		17.61
4	8.5	6		18.84
4	13	6		15.24
8	0	6		12.34
8	4.5	6		12.4
8	8.5	6		12.14
8	13	6		12.35
12	0	6		8.69
12	4.5	6		6.04
12	8.5	6		5.76
12	13	6		8.65
14	0	6		7.67
14	4.5	6		6.67
14	8.5	6		6.63
14	13	6		7.55
16	0	6		7.33
16	4.5	6		6.8
16	8.5	6		6.79
16	13	6		7.28

Cross Panel Data Points

X	Y	Z		Voltage
0	0	0		15.98
0	4.5	0		16.36
0	8.5	0		16.36
0	13	0		15.76
2	0	0		15.62
2	4.5	0		16.38
2	8.5	0		16.62
2	13	0		15.46
4	0	0		14.74
4	4.5	0		16.35
4	8.5	0		16.64
4	13	0		14.51
8	0	0		12.11
8	4.5	0		12.04
8	8.5	0		12.09
8	13	0		11.9
12	0	0		9.17
12	4.5	0		7.58
12	8.5	0		6.99
12	13	0		9.2
14	0	0		8.18
14	4.5	0		7.27
14	8.5	0		7.15
14	13	0		8.14
16	0	0		7.94
16	4.5	0		7.42
16	8.5	0		7.31
16	13	0		7.91
0	0	3		15.87
0	4.5	3		16.44
0	8.5	3		16.37
0	13	3		15.87
2	0	3		15.59
2	4.5	3		16.75
2	8.5	3		16.74
2	13	3		15.54
4	0	3		14.75
4	4.5	3		20.73
4	8.5	3		21.05
4	13	3		14.68
8	0	3		12.35
8	4.5	3		12.26
8	8.5	3		12.44
8	13	3		12.18
12	0	3		9.39
12	4.5	3		3.51
12	8.5	3		3.15
12	13	3		9.18

X	Y	Z		Voltage
14	0	3		8.28
14	4.5	3		7.22
14	8.5	3		7.14
14	13	3		8.22
16	0	3		8.02
16	4.5	3		7.52
16	8.5	3		7.45
16	13	3		7.96
0	0	6		15.9
0	4.5	6		16.39
0	8.5	6		16.36
0	13	6		15.75
2	0	6		15.71
2	4.5	6		16.53
2	8.5	6		16.5
2	13	6		15.44
4	0	6		14.86
4	4.5	6		16.69
4	8.5	6		16.64
4	13	6		14.66
8	0	6		12.53
8	4.5	6		12.68
8	8.5	6		12.24
8	13	6		12.04
12	0	6		9.39
12	4.5	6		7.38
12	8.5	6		7.57
12	13	6		9.35
14	0	6		8.46
14	4.5	6		7.48
14	8.5	6		7.57
14	13	6		8.38
16	0	6		8.19
16	4.5	6		7.63
16	8.5	6		7.68
16	13	6		8.11

Circle Panel Data Point

X	Y	Z		Voltage
0	0	0		13.78
0	4.5	0		13.98
0	8.5	0		14.02
0	13	0		13.97
2	0	0		13.66
2	4.5	0		14.09
2	8.5	0		14.06
2	13	0		13.92
4	0	0		13.35
4	4.5	0		14.03
4	8.5	0		14.19
4	13	0		13.64
8	0	0		12.13
8	4.5	0		12.15
8	8.5	0		12.19
8	13	0		12.36
12	0	0		10.83
12	4.5	0		10.09
12	8.5	0		10.2
12	13	0		10.88
14	0	0		10.61
14	4.5	0		10.23
14	8.5	0		10.25
14	13	0		10.6
16	0	0		10.56
16	4.5	0		10.34
16	8.5	0		10.34
16	13	0		10.49
0	0	3		13.81
0	4.5	3		13.97
0	8.5	3		13.96
0	13	3		13.71
2	0	3		13.7
2	4.5	3		14.04
2	8.5	3		14.08
2	13	3		13.61
4	0	3		13.39
4	4.5	3		14.39
4	8.5	3		14.62
4	13	3		13.41
8	0	3		12.24
8	4.5	3		12.41
8	8.5	3		12.76
8	13	3		12.2
12	0	3		10.72
12	4.5	3		9.66
12	8.5	3		9.61
12	13	3		10.76

X	Y	Z		Voltage
14	0	3		10.5
14	4.5	3		10.12
14	8.5	3		10.12
14	13	3		10.55
16	0	3		10.41
16	4.5	3		10.25
16	8.5	3		10.29
16	13	3		10.48
0	0	6		13.66
0	4.5	6		13.85
0	8.5	6		13.86
0	13	6		13.69
2	0	6		13.58
2	4.5	6		13.98
2	8.5	6		13.96
2	13	6		13.59
4	0	6		13.3
4	4.5	6		14.02
4	8.5	6		14.09
4	13	6		13.31
8	0	6		12.3
8	4.5	6		12.49
8	8.5	6		12.46
8	13	6		12.2
12	0	6		10.76
12	4.5	6		10.18
12	8.5	6		10.27
12	13	6		10.78
14	0	6		10.54
14	4.5	6		10.23
14	8.5	6		10.24
14	13	6		10.57
16	0	6		10.47
16	4.5	6		10.3
16	8.5	6		10.28
16	13	6		10.48

Small Panel Data Points

A.5 MATLAB Scripts

Collection of MATLAB scripts used to generate the voltage and field plots shown in Section 3 and Appendix A.

A.5.1 Top-Down Script

```
%%%%%%%%%%%%%%
Script for generating a top-down graph (map) of the voltage potential at each point.
Variables within function allow for generation of contour map and field map.
%%%%%%%%%%%%%%
function MQP_Test()

    y = [0, 2, 4, 8, 12, 14, 16];
    x = [13, 8.5, 4.5, 0];

    %small panel?
    panel = 1;

    %field testing (1 for on)
    field = 1;

    array = ones(7, 4);

    %TOP LAYER - Square
    %values = [16.84, 17.31, 17.4, 16.72, 16.46, 17.31, 17.52, 16.38, 15.35, 16.94, 17.49, 15.36,
12.61, 12.34, 12.7, 12.46, 8.62, 6.57, 6.8, 8.59, 7.37, 6.15, 6.35, 7.4, 7.02, 6.34, 6.39, 6.97];

    %TOP LAYER - Circle
    %values = [15.9, 16.39, 16.36, 15.75, 15.71, 16.53, 16.5, 15.44, 14.86, 16.69, 16.64, 14.66,
12.53, 12.68, 12.24, 12.04, 9.39, 7.38, 7.57, 9.35, 8.46, 7.48, 7.57, 8.38, 8.19, 7.63, 7.68,
8.11];

    %TOP LAYER - Small
    %values = [13.66, 13.85, 13.86, 13.69, 13.58, 13.98, 13.96, 13.59, 13.3, 14.02, 14.09, 13.31,
12.3, 12.49, 12.46, 12.2, 10.76, 10.18, 10.27, 10.78, 10.54, 10.23, 10.24, 10.57, 10.47, 10.3,
10.28, 10.48];

    %TOP LAYER - Cross
    %values = [16.3, 16.87, 16.91, 16.45, 15.98, 16.99, 17.27, 16.16, 15.06, 17.61, 18.84, 15.24,
12.34, 12.4, 12.14, 12.35, 8.69, 6.04, 5.76, 8.65, 7.67, 6.67, 6.63, 7.55, 7.33, 6.8, 6.79,
7.28];


    %MIDDLE LAYER - Square
    %values = [16.81, 17.38, 17.47, 16.81, 16.49, 17.74, 17.87, 16.47, 15.37, 19.42, 19.9, 15.47,
12.5, 12.64, 12.74, 12.49, 8.66, 3.51, 3.85, 8.44, 7.41, 6.04, 5.85, 7.17, 7.02, 6.38, 6.27,
6.85];

    %MIDDLE LAYER - Circle
    %values = [15.87, 16.44, 16.37, 15.87, 15.59, 16.75, 16.74, 15.54, 14.75, 20.73, 21.05,
14.68, 12.35, 12.26, 12.44, 12.18, 9.39, 3.51, 3.15, 9.18, 8.28, 7.22, 7.14, 8.22, 8.02, 7.52,
7.45, 7.96];

    %MIDDLE LAYER - Small
    %values = [13.81, 13.97, 13.96, 13.71, 13.7, 14.04, 14.08, 13.61, 13.39, 14.39, 14.62, 13.41,
12.24, 12.41, 12.76, 12.2, 10.72, 9.66, 9.61, 10.76, 10.5, 10.12, 10.12, 10.55, 10.41, 10.25,
10.29, 10.48];

    %MIDDLE LAYER - Cross
    %values = [16.36, 16.96, 17, 16.46, 16.01, 17.26, 17.44, 16.17, 15.02, 18.58, 18.46, 15.08,
12.19, 12.71, 12.52, 12.23, 8.7, 4.17, 4.84, 8.73, 7.57, 6.28, 6.33, 7.49, 7.26, 6.65, 6.69,
7.21];
```

```
    %BOTTOM LAYER - Square
    %values = [16.74, 17.35, 17.42, 16.7, 16.33, 17.5, 17.62, 16.32, 15.21, 16.74, 17.12, 15.07,
11.96, 12.03, 12.02, 11.89, 8.83, 6.5, 7.16, 8.64, 7.43, 5.96, 5.98, 7.27, 7, 6.54, 6.22, 6.87];

    %BOTTOM LAYER - Circle
    %values = [15.98, 16.36, 16.36, 15.76, 15.62, 16.38, 16.62, 15.46, 14.74, 16.35, 16.64,
14.51, 12.11, 12.04, 12.09, 11.9, 9.17, 7.58, 6.99, 9.2, 8.18, 7.27, 7.15, 8.14, 7.94, 7.42,
7.31, 7.91];

    %BOTTOM LAYER - Small
    values = [13.78, 13.98, 14.02, 13.97, 13.66, 14.09, 14.06, 13.92, 13.35, 14.03, 14.19, 13.64,
12.13, 12.15, 12.19, 12.36, 10.83, 10.09, 10.2, 10.88, 10.61, 10.23, 10.25, 10.6, 10.56, 10.34,
10.34, 10.49];

    %BOTTOM LAYER - Cross
    %values = [16.36, 16.83, 17.01, 16.41, 16.03, 16.9, 17.23, 16.31, 15.1, 16.3, 17.33, 15.27,
12.02, 11.85, 11.93, 12.08, 9.02, 6.74, 6.58, 8.67, 7.77, 6.65, 6.48, 7.52, 7.44, 6.85, 6.72,
7.26];

    count = 1;
    for n = 1:7
        for i = 1:4
            array(n, i) = values(count);
            count = count + 1;
        end
    end

    newpoints = 45;
    [xq, yq] = meshgrid(linspace(0, 13, newpoints), linspace(0, 16, newpoints));

    arrayQ = griddata(x, y, array, xq, yq, 'linear');

    if(field == 1)
        contour(xq, yq, arrayQ, 20);
    else
        contourf(xq, yq, arrayQ, 20);
    end

    c = colorbar;
    c.Label.String = 'Electric Potential (Volts)';

    xlabel('Y Axis (Width) in Inches');
    ylabel('X Axis (Length) in Inches');

    hold on

        if(panel == 1)
            rectangle('Position', [5.5 10 2 .2], 'FaceColor', 'r');
            rectangle('Position', [5.5 5.8 2 .2], 'FaceColor', 'r');
        else
            rectangle('Position', [4.5 12 4 .2], 'FaceColor', 'r');
            rectangle('Position', [4.5 3.8 4 .2], 'FaceColor', 'r');
        end

    hold off

    if(field == 1)
        [U, V] = gradient(arrayQ * -1, 0.1, 0.1);

        hold on
        quiver(xq, yq, U, V);
        hold off
    end
end
```

A.5.2 Cross-Cut Script

```
%%%%%%%%%%%%%%
Script for generating a cross-cut graph (map) of the voltage potential at each point.
Variables within function allow for generation of contour map and field map.
%%%%%%%%%%%%%%
function MQP_Test()

    x = [0, 4.5, 8.5, 13];
    z = [0, 3, 6];

    %small panel?
    panel = 0;

    array = ones(3, 4);

    % Square Panel

    %ZERO SIDE
    values = [16.74, 17.35, 17.42, 16.7, 16.81, 17.38, 17.47, 16.81, 16.84, 17.31, 17.4, 16.72];

    %TWO SIDE
    %values = [16.33,17.5,17.62,16.32,16.49,17.74,17.87,16.47,16.46,17.31,17.52,16.38];

    %FOUR SIDE
    %values = [15.21,16.74,17.12,15.07,15.37,19.42,19.9,15.47,15.35,16.94,17.49,15.36];

    %EIGHT SIDE
    %values = [11.96,12.03,12.02,11.89,12.5,12.64,12.74,12.49,12.61,12.34,12.7,12.46];

    %TWELVE SIDE
    %values = [8.83,6.5,7.16,8.64,8.66,3.51,3.85,8.44,8.62,6.57,6.8,8.59];

    %FOURTEEN SIDE
    %values = [7.43,5.96,5.98,7.27,7.41,6.04,5.85,7.17,7.37,6.15,6.35,7.4];

    %SIXTEEN SIDE
    %values = [7,6.54,6.22,6.87,7.02,6.38,6.27,6.85,7.02,6.34,6.39,6.97];

    % Circle Panel

    %ZERO SIDE
    %values = [15.98,16.36,16.36,15.76,15.87,16.44,16.37,15.87,15.9,16.39,16.36,15.75];

    %TWO SIDE
    %values = [15.62,16.38,16.62,15.46,15.59,16.75,16.74,15.54,15.71,16.53,16.5,15.44];

    %FOUR SIDE
    %values = [14.74,16.35,16.64,14.51,14.75,20.73,21.05,14.68,14.86,16.69,16.64,14.66];

    %EIGHT SIDE
    %values = [12.11,12.04,12.09,11.9,12.35,12.26,12.44,12.18,12.53,12.68,12.24,12.04];

    %TWELVE SIDE
    %values = [9.17,7.58,6.99,9.2,9.39,3.51,3.15,9.18,9.39,7.38,7.57,9.35];

    %FOURTEEN SIDE
    %values = [8.18,7.27,7.15,8.14,8.28,7.22,7.14,8.22,8.46,7.48,7.57,8.38];

    %SIXTEEN SIDE
    %values = [7.94,7.42,7.31,7.91,8.02,7.52,7.45,7.96,8.19,7.63,7.68,8.11];

    % Small Panel

    %ZERO SIDE
    %values = [13.78,13.98,14.02,13.97,13.81,13.97,13.96,13.71,13.66,13.85,13.86,13.69];

    %TWO SIDE
    %values = [13.66,14.09,14.06,13.92,13.7,14.04,14.08,13.61,13.58,13.98,13.96,13.59];
    %FOUR SIDE
    %values = [13.35,14.03,14.19,13.64,13.39,14.39,14.62,13.41,13.3,14.02,14.09,13.31];

    %EIGHT SIDE
    %values = [12.13,12.15,12.19,12.36,12.24,12.41,12.76,12.2,12.3,12.49,12.46,12.2];

    %TWELVE SIDE
    %values = [10.83,10.09,10.2,10.88,10.72,9.66,9.61,10.76,10.76,10.18,10.27,10.78];

    %FOURTEEN SIDE
    %values = [10.61,10.23,10.25,10.6,10.5,10.12,10.12,10.55,10.54,10.23,10.24,10.57];

    %SIXTEEN SIDE
    %values = [10.56,10.34,10.34,10.49,10.41,10.25,10.29,10.48,10.47,10.3,10.28,10.48];


    % Cross Panel

    %ZERO SIDE
    %values = [16.36,16.83,17.01,16.41,16.36,16.96,17,16.46,16.3,16.87,16.91,16.45];

    %TWO SIDE
    %values = [16.03,16.9,17.23,16.31,16.01,17.26,17.44,16.17,15.98,16.99,17.27,16.16];

    %FOUR SIDE
    %values = [15.1,16.3,17.33,15.27,15.02,18.58,18.46,15.08,15.06,17.61,18.84,15.24];

    %EIGHT SIDE
    %values = [12.02,11.85,11.93,12.08,12.19,12.71,12.52,12.23,12.34,12.4,12.14,12.35];

    %TWELVE SIDE
    %values = [9.02,6.74,6.58,8.67,8.7,4.17,4.84,8.73,8.69,6.04,5.76,8.65];

    %FOURTEEN SIDE
    %values = [7.77,6.65,6.48,7.52,7.57,6.28,6.33,7.49,7.67,6.67,6.63,7.55];

    %SIXTEEN SIDE
    %values = [7.44,6.85,6.72,7.26,7.26,6.65,6.69,7.21,7.33,6.8,6.79,7.28];

    count = 1;
    for n = 1:3
        for i = 1:4
            array(n, i) = values(count);
            count = count + 1;
        end
    end

    newpoints = 50;
    [xq,yq] = meshgrid(linspace(0, 13, newpoints), linspace(0, 6, newpoints));

    arrayQ = griddata(x, z, array, xq, yq, 'linear');

    contourf(xq, yq, arrayQ, 15);

    c = colorbar;
    c.Label.String = 'Electric Potential (Volts)';

    xlabel('Y Axis (Width) in Inches');
    ylabel('Z Axis (Depth) in Inches');

    hold on

    if(panel == 1)
        rectangle('Position', [5.5 2 2 2], 'EdgeColor', 'r');
    else
        rectangle('Position', [4.5 1 4 4], 'EdgeColor', 'r');
    end

    hold off
end
```

A.5.3 Vertical Script

```
%%%%%%%%%%%%%%
Script for generating a vertical (side) graph (map) of the voltage potential at each point.
Variables within function allow for generation of contour map and field map.
%%%%%%%%%%%%%%
function MQP_Test()

    y = [0, 2, 4, 8, 12, 14, 16];
    z = [0, 3, 6];

    %small panel?
    panel = 0;

    array = ones(3, 7);


    % Square Panel

    %ZERO SIDE
    %values =
[16.74,16.33,15.21,11.96,8.83,7.43,7,16.81,16.49,15.37,12.5,8.66,7.41,7.02,16.84,16.46,15.35,12.6
1,8.62,7.37,7.02];

    %FOUR SIDE
    %values =
[17.35,17.5,16.74,12.03,6.5,5.96,6.54,17.38,17.74,19.42,12.64,3.51,6.04,6.38,17.31,17.31,16.94,12
.34,6.57,6.15,6.34];

    %EIGHT SIDE
    %values =
[17.42,17.62,17.12,12.02,7.16,5.98,6.22,17.47,17.87,19.9,12.74,3.85,5.85,6.27,17.4,17.52,17.49,12
.7,6.8,6.35,6.39];

    %THIRTEEN SIDE
    %values =
[16.7,16.32,15.07,11.89,8.64,7.27,6.87,16.81,16.47,15.47,12.49,8.44,7.17,6.85,16.72,16.38,15.36,1
2.46,8.59,7.4,6.97];


    % Circle Panel

    %ZERO SIDE
    %values =
[15.98,15.62,14.74,12.11,9.17,8.18,7.94,15.87,15.59,14.75,12.35,9.39,8.28,8.02,15.9,15.71,14.86,1
2.53,9.39,8.46,8.19];

    %FOUR SIDE
    %values =
[16.36,16.38,16.35,12.04,7.58,7.27,7.42,16.44,16.75,20.73,12.26,3.51,7.22,7.52,16.39,16.53,16.69,
12.68,7.38,7.48,7.63];

    %EIGHT SIDE
    %values =
[16.36,16.62,16.64,12.09,6.99,7.15,7.31,16.37,16.74,21.05,12.44,3.15,7.14,7.45,16.36,16.5,16.64,1
2.24,7.57,7.57,7.68];

    %THIRTEEN SIDE
    %values =
[15.76,15.46,14.51,11.9,9.2,8.14,7.91,15.87,15.54,14.68,12.18,9.18,8.22,7.96,15.75,15.44,14.66,12
.04,9.35,8.38,8.11];


    % Small Panel

    %ZERO SIDE
    %values =
[13.78,13.66,13.35,12.13,10.83,10.61,10.56,13.81,13.7,13.39,12.24,10.72,10.5,10.41,13.66,13.58,13
.3,12.3,10.76,10.54,10.47];

    %FOUR SIDE
    %values =
[13.98,14.09,14.03,12.15,10.09,10.23,10.34,13.97,14.04,14.39,12.41,9.66,10.12,10.25,13.85,13.98,1
4.02,12.49,10.18,10.23,10.3];

    %EIGHT SIDE
    %values =
[14.02,14.06,14.19,12.19,10.2,10.25,10.34,13.96,14.08,14.62,12.76,9.61,10.12,10.29,13.86,13.96,14
.09,12.46,10.27,10.24,10.28];

    %THIRTEEN SIDE
    %values =
[13.97,13.92,13.64,12.36,10.88,10.6,10.49,13.71,13.61,13.41,12.2,10.76,10.55,10.48,13.69,13.59,13
.31,12.2,10.78,10.57,10.48];


    % Cross Panel

    %ZERO SIDE
    %values =
[16.36,16.03,15.1,12.02,9.02,7.77,7.44,16.36,16.01,15.02,12.19,8.7,7.57,7.26,16.3,15.98,15.06,12.
34,8.69,7.67,7.33];

    %FOUR SIDE
    %values =
[16.83,16.9,16.3,11.85,6.74,6.65,6.85,16.96,17.26,18.58,12.71,4.17,6.28,6.65,16.87,16.99,17.61,12
.4,6.04,6.67,6.8];

    %EIGHT SIDE
    %values =
[17.01,17.23,17.33,11.93,6.58,6.48,6.72,17,17.44,18.46,12.52,4.84,6.33,6.69,16.91,17.27,18.84,12.
14,5.76,6.63,6.79];

    %THIRTEEN SIDE
    %values =
[16.41,16.31,15.27,12.08,8.67,7.52,7.26,16.46,16.17,15.08,12.23,8.73,7.49,7.21,16.45,16.16,15.24,
12.35,8.65,7.55,7.28];

    count = 1;
    for n = 1:3
        for i = 1:7
            array(n, i) = values(count);
            count = count + 1;
        end
    end

    newpoints = 45;
    [xq,yq] = meshgrid(linspace(0, 16, newpoints), linspace(0, 6, newpoints));

    arrayQ = griddata(y, z, array, xq, yq, 'linear');

    contourf(xq, yq, arrayQ, 20);

    c = colorbar;
    c.Label.String = 'Electric Potential (Volts)';

    xlabel('X Axis (Length) in Inches');
    ylabel('Z Axis (Depth) in Inches');

    hold on

    if(panel == 1)
        rectangle('Position', [5.8 2 .2 2], 'FaceColor', 'r');
        rectangle('Position', [10 2 .2 2], 'FaceColor', 'r');
    else
        rectangle('Position', [3.8 1 .2 4], 'FaceColor', 'r');
        rectangle('Position', [12 1 .2 4], 'FaceColor', 'r');
    end

    hold off
end
```

A.5.4 System Resistance Script

```
%%%%%%%%%%%%%%
Script for generating calculating the system resistance across each steel panel. Script
adjusts for the change in resistance throughout the panel as you move away from center.
%%%%%%%%%%%%%%

% number of grid points
    N = 9;

% system voltage
    V = 24;

% steel resistivity (ohm*m)
    pS = 6.9E-7;

% water resistivity (ohm*m)
    pW = .2;

%water path resistance
    rW = pW * (0.2032 / 0.0001274);

%panel length (m)
    l = 0.1016;

% path resistance
    paths = zeros(N, N);

% path coordinates
    x = linspace(-l/2, l/2, N);
    y = linspace(-l/2, l/2, N);

% grid
    [xG, yG] = meshgrid(x, y);

% distances
    D = sqrt(xG.^2 + yG.^2);

% steel path resistance
    sR = (D.*pS)/(2.52e-6);

% ====================
% Panel Resistance
% ====================

%     [C, h] = contourf(sR);
%     C = colorbar;
%     C.Label.String = 'Resistance (Ohms)';
%     title('Steel Panel (4" x 4") Resistance');
%     xlabel('Grid Points');
%     ylabel('Grid Points');

% total resisitance in path
    tR = (sR.*2) + rW;

% ====================
% System Resistance
% ====================


%     [C, h] = contourf(tR);
%     C = colorbar;
%     C.Label.String = 'Resistance (Ohms)';
%     title('System Area (4" x 4") Resistance');
%     xlabel('Grid Points (Half-Inch)');
%     ylabel('Grid Points (Half-Inch)');


% ====================
% System Current
% ====================

    bC = V./tR;

    [C, h] = contourf(bC);
    C = colorbar;
    C.Label.String = 'Current (A)';
    title('Panel Current');
    xlabel('Grid Points (Half-Inch)');
    ylabel('Grid Points (Half-Inch)');

    %disp(bC);

    sumBCCol = sum(bC);
    sumBC = sum(sumBCCol);

% 1/R calc (inverse R)
    iR = 1./tR;

% sum of inverse resistances
    sumIRCol = sum(iR);
    sumIR = sum(sumIRCol);

% system resistance
    res = 1/sumIR;

% print total resistance
disp(res);
```

Appendix B - TensorFlow Scripts

B.1 Convert Datatturks to PascalVOC Format

```
##Start Dataturks --> PascalVOC Script
import argparse
import sys
import os
import json
import logging
import requests
from PIL import Image
from io import BytesIO

###################  INSTALLATION NOTE #######################
##############################################################

## pip install requests
## pip install pillow

###############################################################
###############################################################


# enable info logging.
logging.getLogger().setLevel(logging.INFO)

def maybe_download(image_url, image_dir):
    """Download the image if not already exist, return the
location path"""
    fileName = image_url.split("/")[-1]
    filePath = os.path.join(image_dir, fileName)

    # Trying to convert to jpg before

    #################################

    if os.path.exists(filePath):
        print('*****Found an image that exists: ' + filePath)
        return filePath

    if filePath.lower().endswith(".png"):  # png image
        print('Found a png image --> ' + filePath)
        try:
            response = requests.get(image_url)
            if response.status_code == 200:
                img = Image.open(BytesIO(response.content))
                m_img = Image.new("RGB", img.size, (255, 255,
255))
                m_img.paste(img, (0, 0), img)
                filePath = filePath[:-3] + 'jpg'
                m_img.save(filePath, quality=95)

                img.close()
                return filePath
            else:
                raise ValueError("Not a 200 response")
        except Exception as e:
            logging.exception("Failed to download image at " +
image_url + " \n" + str(e) + "\nignoring....")
            raise e
    elif filePath.lower().endswith(".jpg"):  # jpg image
        # else download the image
        print('Found a jpg image --> ' + filePath)
        try:
            response = requests.get(image_url)
            if response.status_code == 200:
                with open(filePath, 'wb') as f:
                    f.write(response.content)
                    return filePath
            else:
                raise ValueError("Not a 200 response")
        except Exception as e:
            logging.exception("Failed to download image at " +
image_url + " \n" + str(e) + "\nignoring....")
            raise e
    else:
        print('$$$$$Not a valid file --> ' + filePath)
        logging.exception("Failed to download image at " +
image_url + " \n" + str(e) + "\nignoring....")
        raise ValueError("Not a jpg or png")

def get_xml_for_bbx(bbx_label, bbx_data, width, height):

    if len(bbx_data['points']) == 4:
        #Regular BBX has 4 points of the rectangle.
        xmin = width*min(bbx_data['points'][0][0],
bbx_data['points'][1][0], bbx_data['points'][2][0],
bbx_data['points'][3][0])
        ymin = height * min(bbx_data['points'][0][1],
bbx_data['points'][1][1], bbx_data['points'][2][1],
                           bbx_data['points'][3][1])
        xmax = width * max(bbx_data['points'][0][0],
bbx_data['points'][1][0], bbx_data['points'][2][0],
                           bbx_data['points'][3][0])
        ymax = height * max(bbx_data['points'][0][1],
bbx_data['points'][1][1], bbx_data['points'][2][1],
                           bbx_data['points'][3][1])

    else:
        #OCR BBX format has 'x','y' in one point.
        # We store the left top and right bottom as point '0'
and point '1'
        xmin = int(bbx_data['points'][0]['x']*width)
        ymin = int(bbx_data['points'][0]['y']*height)
        xmax = int(bbx_data['points'][1]['x']*width)
        ymax = int(bbx_data['points'][1]['y']*height)

    xml = "<object>\n"
    xml = xml + "\t<name>" + bbx_label + "</name>\n"
    xml = xml + "\t<pose>Unspecified</pose>\n"
    xml = xml + "\t<truncated>Unspecified</truncated>\n"
    xml = xml + "\t<difficult>Unspecified</difficult>\n"
    xml = xml + "\t<occluded>Unspecified</occluded>\n"
    xml = xml + "\t<bndbox>\n"
    xml = xml +     "\t\t<xmin>" + str(xmin) + "</xmin>\n"
    xml = xml +     "\t\t<xmax>" + str(xmax) + "</xmax>\n"
    xml = xml +     "\t\t<ymin>" + str(ymin) + "</ymin>\n"
    xml = xml +     "\t\t<ymax>" + str(ymax) + "</ymax>\n"
    xml = xml + "\t</bndbox>\n"
    xml = xml + "</object>\n"
    return xml


def convert_to_PascalVOC(dataturks_labeled_item, image_dir,
xml_out_dir):

    """Convert a dataturks labeled item to pascalVOCXML string.
      Args:
        dataturks_labeled_item: JSON of one labeled image from
dataturks.
        image_dir: Path to  directory to downloaded images (or a
directory already having the images downloaded).
        xml_out_dir: Path to the dir where the xml needs to be
written.
      Returns:
        None.
      Raises:
        None.
        """
    try:
        data = json.loads(dataturks_labeled_item)
        if len(data['annotation']) == 0:
            logging.info("Ignoring Skipped Item");
            return False;

        width = data['annotation'][0]['imageWidth']
        height = data['annotation'][0]['imageHeight']
        image_url = data['content']

        filePath = maybe_download(image_url, image_dir)

        with Image.open(filePath) as img:
            width, height = img.size

        fileName = filePath.split("/")[-1]
        image_dir_folder_Name = image_dir.split("/")[-1]


        xml = "<annotation>\n<folder>images</folder>\n"
        xml = xml + "<filename>" + fileName +"</filename>\n"
        xml = xml + "<path>" + filePath +"</path>\n"
        xml = xml +
"<source>\n\t<database>Unknown</database>\n</source>\n"
        xml = xml + "<size>\n"
        xml = xml +     "\t<width>" + str(width) + "</width>\n"
        xml = xml +    "\t<height>" + str(height) +
"</height>\n"
        xml = xml +    "\t<depth>Unspecified</depth>\n"
        xml = xml +  "</size>\n"
        xml = xml + "<segmented>Unspecified</segmented>\n"

        for bbx in data['annotation']:
            if not bbx:
                continue;
            #Pascal VOC only supports rectangles.
            if "shape" in bbx and bbx["shape"] != "rectangle":
                continue;

            bbx_labels = bbx['label']
            #handle both list of labels or a single label.
            if not isinstance(bbx_labels, list):
                bbx_labels = [bbx_labels]

            for bbx_label in bbx_labels:
                xml = xml + get_xml_for_bbx(bbx_label, bbx,
width, height)

        xml = xml + "</annotation>"

        fileName = fileName[:-4]

        #output to a file.
        xmlFilePath = os.path.join(xml_out_dir, fileName +
".xml")
        with open(xmlFilePath, 'w') as f:
            f.write(xml)
        return True
    except Exception as e:
        logging.exception("Unable to process item " +
dataturks_labeled_item + "\n" + "error = "  + str(e))
        return False

def main():
    #make sure everything is setup.
    if (not os.path.isdir(image_download_dir)):
        logging.exception("Please specify a valid directory path
to download images, " + image_download_dir + " doesn't exist")
        return
    if (not os.path.isdir(pascal_voc_xml_dir)):
        logging.exception("Please specify a valid directory path
to write Pascal VOC xml files, " + pascal_voc_xml_dir + "
doesn't exist")
        return
    if (not os.path.exists(dataturks_JSON_FilePath)):
        logging.exception(
            "Please specify a valid path to dataturks JSON
output file, " + dataturks_JSON_FilePath + " doesn't exist")
        return

    lines = []
    with open(dataturks_JSON_FilePath, 'r') as f:
        lines = f.readlines()

    if (not lines or len(lines) == 0):
        logging.exception(
            "Please specify a valid path to dataturks JSON
output file, " + dataturks_JSON_FilePath + " is empty")
        return

    count = 0;
    success = 0
    for line in lines:
        status = convert_to_PascalVOC(line, image_download_dir,
pascal_voc_xml_dir)
        if (status):
            success = success + 1

        count+=1;
        if (count % 10 == 0):
            logging.info(str(count) + " items done ...")

    logging.info("Completed: " + str(success) + " items done, "
+ str(len(lines) - success)  + " items ignored due to errors or
for being skipped items.")


def create_arg_parser():
    """"Creates and returns the ArgumentParser object."""

    parser = argparse.ArgumentParser(description='Converts
Dataturks output JSON file for Image bounding box to Pascal VOC
format.')
    parser.add_argument('dataturks_JSON_FilePath',
                    help='Path to the JSON file downloaded from
Dataturks.')
    parser.add_argument('image_download_dir',
                    help='Path to the directory where images
will be dowloaded (if not already found in the directory).')
    parser.add_argument('pascal_voc_xml_dir',
                        help='Path to the directory where Pascal
VOC XML files will be stored.')
    return parser

if __name__ == '__main__':
    arg_parser = create_arg_parser()
    parsed_args = arg_parser.parse_args(sys.argv[1:])
    global dataturks_JSON_FilePath
    global image_download_dir
    global pascal_voc_xml_dir

    #setup global paths needed accross the script.
    dataturks_JSON_FilePath =
parsed_args.dataturks_JSON_FilePath
    image_download_dir = parsed_args.image_download_dir
    pascal_voc_xml_dir = parsed_args.pascal_voc_xml_dir
    main()

##End Dataturks --> PascalVOC Script
```

B.2 Convert To JPG

##Start JPG conversion script

```
from PIL import Imagefrom os import listdirfrom os.path import
splitexttarget_directory = '.'target = '.jpg'for file in
listdir(target_directory):filename, extension = splitext(file)try:    if extension
not in ['.py', target]:        im = Image.open(filename + extension)
im.save(filename + target)except OSError:    print('Cannot convert %s' % file)
```

##End JPG conversion script

B.3 XML to CSV

```
# Taken from https://github.com/datitran/raccoon_dataset
# Modified by
# https://pythonprogramming.net/creating-tfrecord-files-
tensorflow-object-detection-api-tutorial/
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET


def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        print (xml_file)
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                     int(root.find('size')[0].text),
                     int(root.find('size')[1].text),
                     member[0].text,
                     int(float(member[5][0].text)),
                     int(float(member[5][1].text)),
                     int(float(member[5][2].text)),
                     int(float(member[5][3].text))
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height',
                   'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df


def main():
    for directory in ['train', 'test']:
        image_path = os.path.join(os.getcwd(),
'images/{}'.format(directory))
        xml_df = xml_to_csv(image_path)
        print (image_path)
        xml_df.to_csv('data/{}_labels.csv'.format(directory),
index=None)
        print('Successfully converted xml to csv.')

if __name__ == '__main__':
    main()
```

B.4 Create TFRecord from CSV and Images

```
"""
Usage:
  # From tensorflow/models/
  # Create train data:
  python generate_tfrecord.py --csv_input=data/train_labels.csv
--output_path=train.record

  # Create test data:
  python generate_tfrecord.py --csv_input=data/test_labels.csv
--output_path=test.record
"""
from __future__ import division
from __future__ import print_function
from __future__ import absolute_import

import os
import io
import pandas as pd
import tensorflow as tf

from PIL import Image
from object_detection.utils import dataset_util
from collections import namedtuple, OrderedDict

flags = tf.app.flags
flags.DEFINE_string('csv_input', '', 'Path to the CSV input')
flags.DEFINE_string('output_path', '', 'Path to output
TFRecord')
flags.DEFINE_string('image_dir', '', 'Path to images')
FLAGS = flags.FLAGS


# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'lionfish':
        return 1
    elif row_label == 'shark':
        return 2
    elif row_label == 'human diver':
        return 3
    elif row_label == 'sea urchin':
        return 4
    else:
        None
```

```
def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in
zip(gb.groups.keys(), gb.groups)]


def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path,
'{}'.format(group.filename)), 'rb') as fid:
        encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size

    filename = group.filename.encode('utf8')
    image_format = b'jpg'
    xmins = []
    xmaxs = []
    ymins = []
    ymaxs = []
    classes_text = []
    classes = []

    for index, row in group.object.iterrows():
        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)

        # print('class_text --> ' + row['class'])

        classes_text.append(row['class'].encode('utf8'))
        classes.append(class_text_to_int(row['class']))

    tf_example =
tf.train.Example(features=tf.train.Features(feature={
        'image/height': dataset_util.int64_feature(height),
        'image/width': dataset_util.int64_feature(width),
        'image/filename': dataset_util.bytes_feature(filename),
        'image/source_id': dataset_util.bytes_feature(filename),
        'image/encoded':
dataset_util.bytes_feature(encoded_jpg),
        'image/format':
dataset_util.bytes_feature(image_format),
        'image/object/bbox/xmin':
dataset_util.float_list_feature(xmins),
        'image/object/bbox/xmax':
dataset_util.float_list_feature(xmaxs),
        'image/object/bbox/ymin':
dataset_util.float_list_feature(ymins),
        'image/object/bbox/ymax':
dataset_util.float_list_feature(ymaxs),
        'image/object/class/text':
dataset_util.bytes_list_feature(classes_text),
        'image/object/class/label':
dataset_util.int64_list_feature(classes),
    }))
    return tf_example


def main(_):
    writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
    path = os.path.join(FLAGS.image_dir)
    examples = pd.read_csv(FLAGS.csv_input)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())

    writer.close()
    output_path = os.path.join(os.getcwd(), FLAGS.output_path)
    print('Successfully created the TFRecords:
{}'.format(output_path))


if __name__ == '__main__':
    tf.app.run()
```

B.5 Commands for Training on GCP

```
//--------------------General
export PYTHONPATH=$PYTHONPATH:pwd:pwd/slim


//--------------------TPU
#Follow tutorial on:
#    "https://medium.com/tensorflow/training-and-serving-a-
realtime-mobile-object-detector-in-30-minutes-with-cloud-tpus-
b78971cf1193"


export PROJECT="nomadic-zoo-234019    "
export YOUR_GCS_BUCKET="gs://lionfish-mqp"


#Get TPU service account
curl -H "Authorization: Bearer $(gcloud auth print-access-
token)"  \
    https://ml.googleapis.com/v1/projects/${PROJECT}:getConfig


export TPU_ACCOUNT="service-68776937774@cloud-
tpu.iam.gserviceaccount.com"


gcloud projects add-iam-policy-binding $PROJECT  \
    --member serviceAccount:$TPU_ACCOUNT --role
roles/ml.serviceAgent


#Run training job on TPU
gcloud ml-engine jobs submit training
`whoami`_object_detection_`date +%s` \
--job-dir=gs://lionfish-mqp/train \
--packages dist/object_detection-0.1.tar.gz,slim/dist/slim-
0.1.tar.gz,/tmp/pycocotools/pycocotools-2.0.tar.gz \
--module-name object_detection.model_tpu_main \
--runtime-version 1.12 \
--scale-tier BASIC_TPU \
--region us-central1 \
-- \
--model_dir=gs://lionfish-mqp/train \
--tpu_zone us-central1 \
--pipeline_config_path=gs://lionfish-mqp/data/pipeline.config
```

```
#Run eval job on TPU
gcloud ml-engine jobs submit training
`whoami`_object_detection_eval_validation_`date +%s` \
--job-dir=gs://lionfish-mqp/train \
--packages dist/object_detection-0.1.tar.gz,slim/dist/slim-
0.1.tar.gz,/tmp/pycocotools/pycocotools-2.0.tar.gz \
--module-name object_detection.model_main \
--runtime-version 1.12 \
--scale-tier BASIC_GPU \
--region us-central1 \
-- \
--model_dir=gs://lionfish-mqp/train \
--pipeline_config_path=gs://lionfish-mqp/data/pipeline.config \
--checkpoint_dir=gs://lionfish-mqp/train


#View the status of the training on GCP
tensorboard --logdir=gs://lionfish-mqp/train


//--------------------NON TPU
#Follow tutorial on:
#    "https://cloud.google.com/blog/products/gcp/training-an-
object-detector-using-cloud-machine-learning-engine"
export PROJECT="nomadic-zoo-234019"
export YOUR_GCS_BUCKET="gs://lionfish-mqp"


# From tensorflow/models/research/
bash
object_detection/dataset_tools/create_pycocotools_package.sh
/tmp/pycocotools
python setup.py sdist
(cd slim && python setup.py sdist)


######NOTE: When you are modifying the pipeline.config files,
make sure to change the num_classes paramenter along with the
other gs file locations


#Command to train on GCP
gcloud ml-engine jobs submit training
`whoami`_object_detection_`date +%s` \
--job-dir=gs://lionfish-mqp/train \
--packages dist/object_detection-0.1.tar.gz,slim/dist/slim-
0.1.tar.gz,/tmp/pycocotools/pycocotools-2.0.tar.gz \
--module-name object_detection.model_main \
--region us-central1 \
--config object_detection/samples/cloud/cloud.yml \
-- \
--model_dir=gs://lionfish-mqp/train \
--pipeline_config_path=gs://lionfish-mqp/data/pipeline.config


#Command to eval training
gcloud ml-engine jobs submit training
`whoami`_object_detection_eval_validation_`date +%s` \
--job-dir=gs://lionfish-mqp/train \
--packages dist/object_detection-0.1.tar.gz,slim/dist/slim-
0.1.tar.gz,/tmp/pycocotools/pycocotools-2.0.tar.gz \
--module-name object_detection.model_main \
--region us-central1 \
--config object_detection/samples/cloud/cloud.yml \
-- \
--model_dir=gs://lionfish-mqp/train \
--pipeline_config_path=gs://lionfish-mqp/data/pipeline.config \
--checkpoint_dir=gs://lionfish-mqp/train


#View progress of training in browser
tensorboard --logdir=gs://lionfish-mqp/train


#Convert graph.pbtxt -> graph.pb (convert to binart type) this
requires modifying
#    the script to get the correct local path to graph.pbtxt
python pbtxt_to_pb.py

#Convert the graph.pb to a frozen graph
python export_inference_graph.py \
    --input_type image_tensor \
    --pipeline_config_path video/video6/pipeline.config \
    --trained_checkpoint_prefix video/video6/model.ckpt-200014 \
    --output_directory video/video6

#Run the video processing
#    Note: You must modify the script to point to the correct
checkpoint, label_map,
#    videos, and the resolution of the output video matches the
input video
python video_processing.py
```