Conditioning Planaria:
Device Design Based on an Autonomous, Large-Scale Parallel Approach

by

Dawn M. Wendell

Submitted to the Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

June 2004

© 2004 Dawn M. Wendell. All rights reserved.
The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part.

Signature of Author..
Department of Mechanical Engineering
May 7, 2004

Certified by..
Ian W. Hunter
Hastopoulos Professor of Mechanical Engineering and Professor of BioEngineering
Thesis Supervisor

Accepted by..
Ernest G. Cravahlo
Professor of Mechanical Engineering
Chairman, Undergraduate Thesis Committee

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
OCT 2 8 2004
LIBRARIES

ARCHIVES

Conditioning Planaria:
Device Design Based on an Autonomous, Large-Scale Parallel Approach

by

Dawn M. Wendell

Submitted to the Department of Mechanical Engineering
on May 7, 2004 in Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science in Mechanical Engineering

ABSTRACT

Current behavioral research is conducted on planaria that have been conditioned manually, one at a time by a person. In an attempt to instrument at an organism level, a design for an environment that automatically conditions multiple planaria in parallel was produced. This design consisted of a testing chamber that could stimulate the planaria using electrical shock and light. A computer program was also written to automatically record the results of the experiments for later analysis by researchers. This design was tested and the results were inconclusive based on technical issues with the experimental procedure. Further research is necessary to determine the validity of this device's ability to condition planaria.

Thesis Supervisor: Ian W. Hunter
Title: Hastopoulos Professor of Mechanical Engineering and Professor of BioEngineering

Acknowledgements

First and foremost, I thank Professor Ian Hunter for giving me the opportunity to work in the BioInstrumentation Lab, and also for guidance and support on my project. This project has showed me the amazing, challenging, and yet fun aspects of research. Also, much thanks to the members of the BioInstrumentation Lab for their help answering all of my questions and making me feel welcome in the lab.

To my friends, thanks for making me smile and showing me on a daily basis that this school is filled with amazing people.

And of course, thanks to my family, especially my parents, who have always supported everything I've ever done.

Table of Contents

1 Introduction

Recent advances in molecular biology and medicine have led to an accelerated pace of discovery for treatments for various diseases. Also, new technology has led to high-throughput screening methods that allow companies to quickly test thousands of compounds as potential new drugs [1, 2]. However, these methods only work for small molecular-based diseases based on proteins or small molecules. In order to further advance the search for new pharmaceuticals for diseases that are manifest only at the organism level, there is a need for a method that allows for high-throughput screening of organisms.

One of the most difficult areas to study is memory. Diseases that affect memory, such as Alzheimer's, are not understood on a molecular basis. Behavioral research is studied on an organism level [3]. Take for example Dr. Michael Levine's laboratory at the Forsyth Institute in Boston [4]. Memory is studied using planaria, or common flatworms. These planaria are given a memory by "training" them to either turn a certain direction when they approach a choice point in a channel or to associate one benign stimulus with a physically painful one. Each of the planaria is trained individually by the researchers. This method is extremely time-consuming and also incorporates an element of human error and inconsistency into the experimentation. Similar methods have been used since such memory tests began in the 1950s [5, 6, 7]. Advances in technology over the past half-century should make it possible to invent a better method for training the planaria consistently and automatically.

This thesis explores the possibilities for automating the planaria conditioning process, allowing conditioning to be performed on a large-scale basis with several planaria being trained in parallel. Functional requirements are described and various designs are tested, culminating in an experiment to condition planaria using the final design. Results are analyzed and future research directions are discussed.

2 Background

In order to understand the complexities surrounding the conditioning of planaria, it is necessary to have a comprehension of the psychology of conditioning as well as a background in the usage of planaria as experimental organisms.

2.1 Psychology of Conditioning

There are two general categories of conditioning: classical conditioning and instrumental conditioning.

2.1.1 Classical Conditioning

Classical conditioning, also known as Pavlovian conditioning, involves associating a conditioned reflex to a different stimulus than usual. Early research performed by Ivan Pavlov showed that dogs would salivate reliably at the sound of a bell that had previously always signaled the arrival of food into their mouths [8]. In this case, a conditioned stimulus (the bell) had been associated with a conditioned response (salivation), as opposed to the unconditioned stimulus of the food placed in the dog's mouth eliciting an unconditioned response of salivation. The dog had learned to associate the bell with the arrival of food, and therefore the food was no longer necessary to obtain the response of salivation.

2.1.2 Instrumental Conditioning

Instrumental conditioning, also known as Operant or Skinner's conditioning, is different than classical conditioning because it requires the subject to perform some action in order to produce a beneficial effect, such as a reward of food or water [8]. This type of conditioning was first observed by Edward Lee Thorndike. He made a puzzle box in which there was a lever that opened the door of the box. He would put a hungry cat inside the box, place a dish of food outside the box, and then time how long until the cat pressed the lever and was able to leave the box and eat the food. Thorndike showed that the variety of responses that a cat expressed when first placed in the box would slowly be narrowed to only the response that successfully opened the box door after many trials. The cat was learning to perform an action in order to get the desired response (in this case, food).

Burrhus Skinner furthered Thorndike's research by creating a Skinner's Box in which a lever or other mechanism could be operated by the subject animal, leading to some sort of reward such as a food pellet or drop of water [8]. This invention allowed the subject to remain in the cage until the trial was over and also allowed it to respond as often as it desired. A decrease in time between responses showed that the animal was learning.

2.2 Planaria as Experimental Organisms

Planaria are excellent experimental organisms because they are small (5-25 mm in length, 1 m wide, 0.1 mm thick) and easy to keep in culture [Figure 2]. They survive well in aerated pond water, fed once a week with liver after which the water is changed [9]. They are the simplest animal with bilateral eyespots [10] and are negatively phototaxic [9]. Planaria are considered good experimental subjects for memory studies because they are the first organisms to show true synaptic nervous transmission and definite encephalization [5]. (For a phylogenic tree, see Figure 1.)

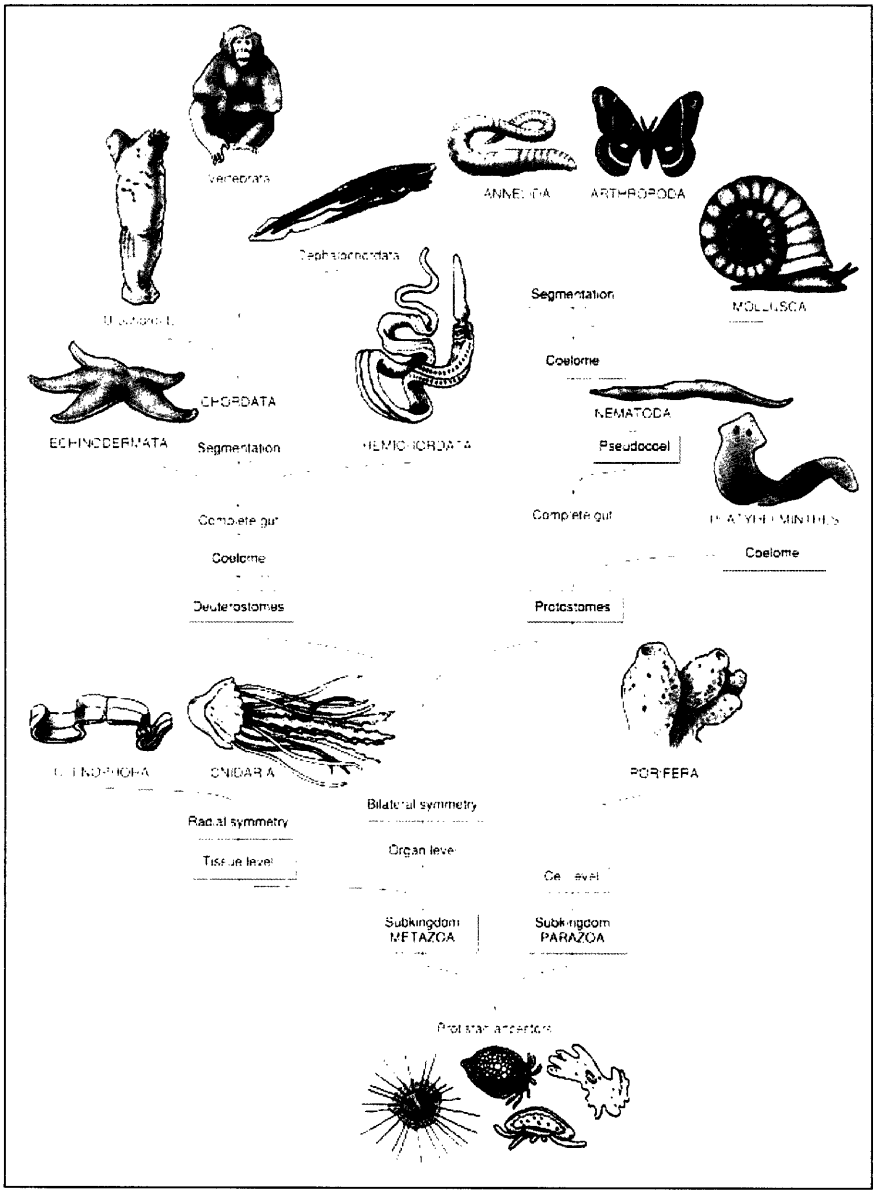


Figure 1: Phylogenic tree showing the planaria as belonging to the class *Platyheminthes* (taken from http://dragon.seowon.ac.kr/~bioedu/bio/ch30.htm)

However, planaria are all hermaphrodites and can reproduce sexually or asexually [9]. They "do not have a sequestered germ line," [5] making them ineffective for traditional genetic experimentation.

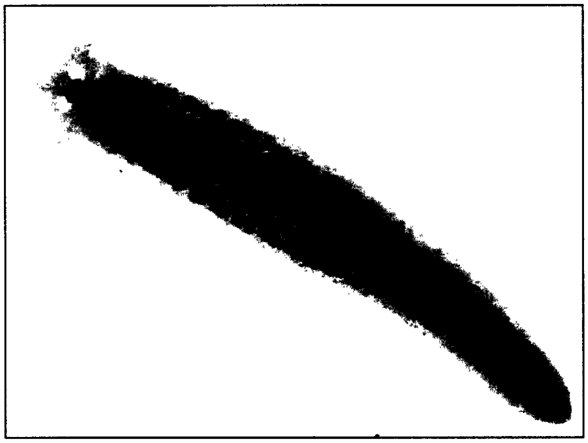


Figure 2: A planarian, *dugesia japonica*
(taken from http://www.luc.edu/depts/biology/111/planaria.htm)

2.3 Previous Behavioral Memory Experiments Involving Planaria

Research into the ability of planaria to "learn" began in the late 1920s, with the first paper about planaria's conditioned responses being published in 1929 [6]. By 1955, experiments performed by Thompson and McConnell proved that planaria were capable of conditioned responses [5]. (Previous experiments had lacked all the necessary control groups). These experiments involved running the planaria in shallow, water-filled troughs. (Modern commercial examples for high-school biology experiments are shown in Figure 3).

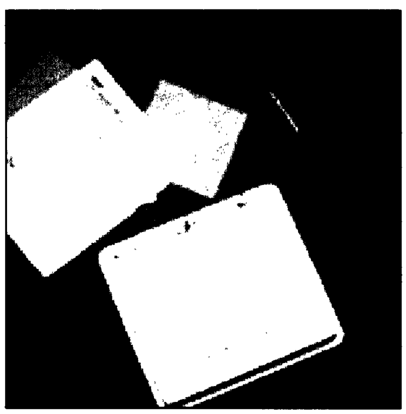


Figure 3: Modern examples of dishes for planaria conditioning experiments, commercially available from Wards Natural Science Establishment [9].

In 1962, Best and Rubinstein published results showing that planaria were capable of instrumental conditioning and they also noted that the planaria exhibited behaviors usually attributed to much higher organisms, such as "vicarious trial-and-error behavior" when they were at a choice point [6].

Once the learning abilities of the planaria had been proven, experiments were performed that seemed to suggest that planaria could learn faster if they cannibalized planaria who had already been conditioned. However, doubt was cast on these experiments by other scientists [3], [11]. Planaria slowly lost favor with biologists, and other organisms came to the forefront as genetic research expanded [12].

The importance of planaria has re-emerged recently with their possible uses as screening organisms. A planarian model of Parkinson's Disease has been established using 1-methyl-4-phenyl-1,2,3,6-tetrahydropyridine (MPTP) [13] and planaria have been used for screening of chemical carcinogens [14]. The planaria will most likely be a very important organism in the future of diseases that manifest themselves at the level of the organism.

3 Planaria Training Environment Design

The goal of the Planaria Training Environment is to automatically condition numerous planaria in parallel, easily and repeatably. A design to satisfy these requirements can be divided into different functional parts. The mechanical portion of the device is where the planaria is placed. The electronics stimulate the planaria and provide environmental cues, and the computer programming controls the stimulation based on data from a camera that is viewing the planaria. Data associated with the planaria and their responses are recorded for analysis by the researchers.

3.1 Mechanical Design of Apparatus

The mechanical portion of the apparatus contains the planaria physically, allows for a steady influx of clean water and an outflow of waste water, and also provides a method of electrical stimulation.

3.1.1 Proof Of Concept: The Delrin Design

The first mechanical design was a chamber with a removable top and bottom section [Figure 4]. The worm was contained in a cylinder capped with mesh disks. The disks could be electrified to create a potential through the water. Water flow was maintained through the cylinder by channels above the top mesh and below the bottom mesh (so that the worm was contained by the mesh and therefore unable to crawl into the water channels).

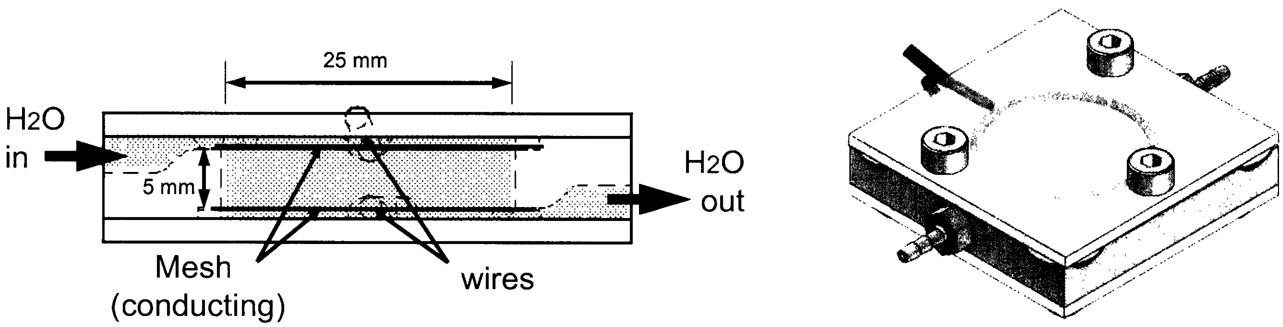


Figure 4: The first proof-of-concept design was a cylinder that contained the worm between two sheets of mesh that could be electrified via two attached wires. Water flowed through the device through tubing but the worm was contained by the mesh. (Images from SolidEdge [15].)

A prototype of the design was built [Figure 5]. The chamber was milled out of acetal using a HAAS VF-OE milling machine [16]. Stainless steel mesh was cut with Electrical Discharge Machining using a Charmilles Technologies AG Robofil 1020SI [17]. (Electrial Discharge Machining or wire EDM uses electrical discharges between the workpiece and a wire to erode the material). Silicon c-rings used to hold the mesh in place on the top and bottom of the chamber were made using an Epilog 45 TT [18], a laser-cutting machine, and were held in place by two sheets of acrylic that were sandwiched on the top and bottom of the acetal chamber. The acrylic was bolted to the acetal to hold the device together. The water inlets and outlets were drilled and tapped holes into which hose fittings were screwed. The parts of the channels that directed the water towards the chamber were surface-milled to provide smooth water flow.

The device was tested by assembling it and flowing water through it using a 5 mL syringe. The resistance across the leads was measured and found to be on the order of 15 kΩ. Leakage was observed around the entire device, so it was decided that the next device would need to specifically address the sealing of the water into the device.

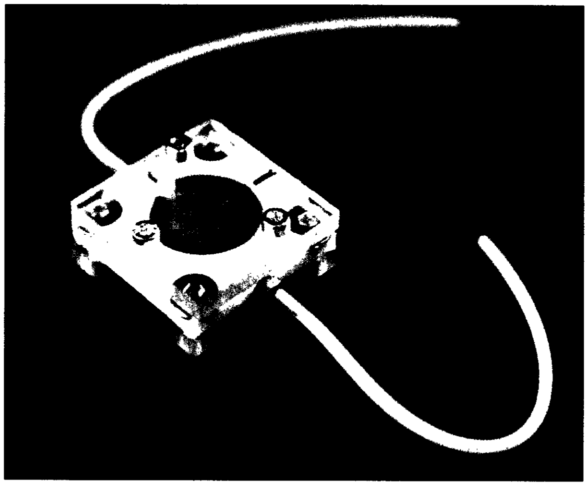


Figure 5: The completed initial prototype, assembled and prepared for testing.

3.1.2 The SLA Flat Inlet Design

The next design was created to alleviate some of the problems discovered with the initial prototype, as well as show the opportunities for creating a device with multiple chambers running in parallel. However, some of the limitations on the original design were based on the manufacturing techniques used. In order to have more flexibility in the design, the use of a novel manufacturing technique was employed. The BioInstrumentation Lab has a 3D Systems Si^2 Stereo-Lithography Apparatus (SLA) machine in the machine shop [19]. This machine creates parts from an epoxy resin by curing the liquid resin with an ultraviolet (UV) laser. It builds the part in layers of 50 μm thick and therefore allows for the quick manufacture of many exciting geometries not available with standard machining, such as enclosed surfaces and long, curved holes. In order to create the geometry necessary for a multi-chamber design, the SLA was used to create the middle part of the device (the acetal portion of the previous design). Also, commercial o-rings were added to attempt to stop the leakage problems that were previously observed. In all other respects, the design was principally the same as the initial prototype.

When adding the additional chambers, it was decided that each chamber needed its own water inlet, but all the chambers could share an outlet. The rational behind this was that at some point in the future, it might be desirable to be able to pump different solutions into each chamber during the training process. However, the water would not be recycled after use because of the waste products from the planaria, so all of the waste streams could be combined.

When the design was machined, assembled and tested, a few issues arose. First, although the commercial o-rings helped the leakage, they failed when the water was flowing through the chambers at a moderate velocity of ~2 mm/s. Even with no mesh or other parts of the system present to impede water flow, the chambers still leaked when water was pumped in at that velocity from a set of syringes [Figure 6].

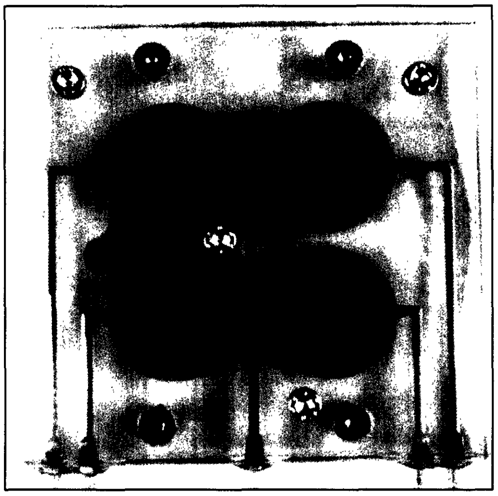


Figure 6: Water dyed with blue food coloring, flow velocity ~2 m/s. Note the leakage of water between the chambers on the left side of the device.

However, when the water velocity was on the order of 0.5 mm/sec, the chambers did not leak, as shown in Figure 7, a screen capture of a movie showing the filling of the chambers.

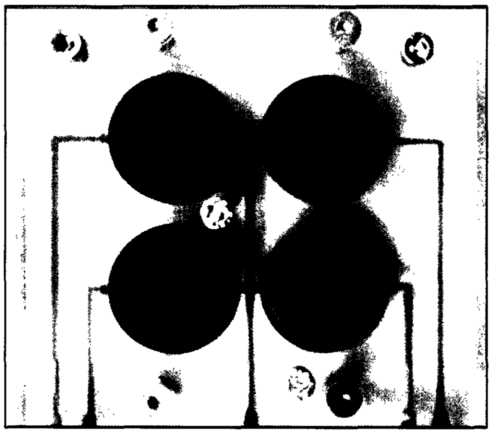


Figure 7: Water dyed with blue food coloring, flow velocity ~0.5 mm/s.

This design was given to the researchers at the Forsyth Institute for testing with planaria. The planaria that were placed in the device had an adverse reaction to epoxy resin used by the SLA. To solve this problem, the device was coated with parylene C, an enert polymer, using a Model 3000 Lab Top Coater, manufactured by Para Tech Coating Inc [20]. Other concerns noted by the researchers involved the ease-of-use of the design. The numerous gaskets, o-rings, and screws holding the device together made it difficult to quickly assemble the device and put planaria into it.

Another issue with this overall design became apparent when a solid-model of a large-scale grid of chambers was created [Figure 8]. This showed that there was much wasted space due to the large numbers of inlets that were all required to be on the same side of the device.

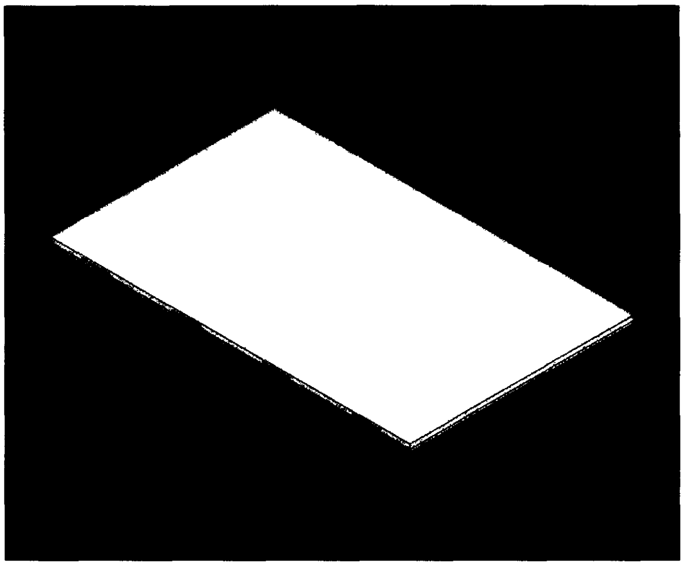


Figure 8: The design for a 64-chamber apparatus. The wasted space due to the large numbers of inlets and outlets was a concern. (Image created in SolidEdge [15].)

It was decided that this design was not the most efficient, so a new design was necessary.

3.1.3 The SLA Vertical Inlet Design

In order to conserve space between the chambers, a new design where the inlets were stacked on top of each other was devised [Figure 9].

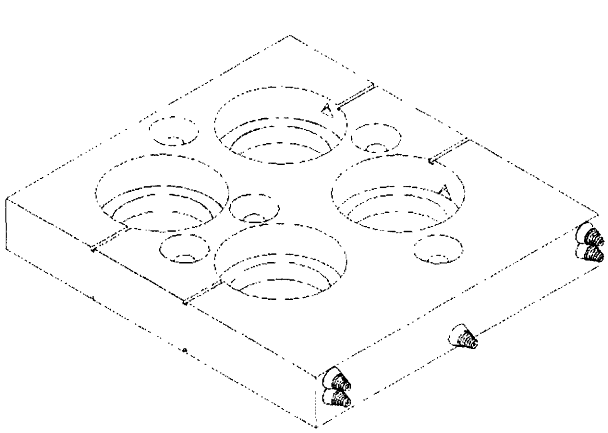


Figure 9: The vertical inlet design. The inlets for a row of chambers were stacked on top of each other to conserve space. (Image created in SolidEdge [15].)

This design was quickly prototyped using the SLA and tested. It performed similarly to the previous SLA design; leakage was still an issue, and it was difficult to assemble and use. In addition, the added height in the chambers increased the difficulty of filling the chambers with water and getting all the air bubbles out. These air bubbles were especially noticeable when the chamber was illuminated with the infrared LEDs and viewed with the camera. [Figure 10].

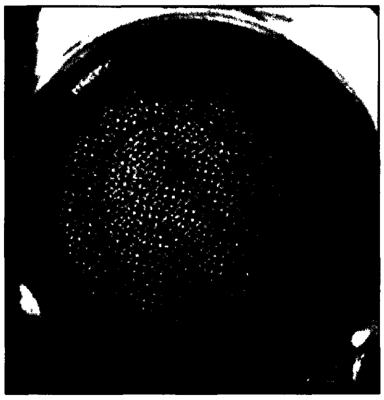


Figure 10: Infrared-illuminated chamber, viewed with an USB2 Orange Micro iBOT camera [21]. The large circle in the center is an air bubble and the dark line on the right is the planaria.

These issues were significant enough that I decided that a completely new approach was necessary. The search for a drastically new idea for the design began.

3.1.4 Other possibilities: Laminated Acrylic design

The next concept examined was a laminated acrylic design. This would consist of multiple layers of acrylic plastic that would be bonded together thermally or chemically to create the inlets, chambers, and other features of the training environment. The most noted benefit was that the bonding of all of the layers together would minimize leaks, however, it was still necessary to be able to disassemble the device to input the planaria and for cleaning, and this meant that the design would still need the numerous seals that were the cause of the leakage in previous designs. The seals would also be potential problem areas for air bubbles. It was decided that this design change would not significantly solve the SLA designs' problems.

3.1.5 Shifting Perspective: The Submerged Capsule Design

Despite the numerous different iterations, the continuing issues with water leakage and ease of use prompted renewed brainstorming about the basic design of the training chamber. A new idea was devised that took a fundamentally-different perspective on the concept. Instead of the water being tightly contained around the chamber and limited to the area close to the mesh, a capsule was designed that would be submerged in water [Figure 11].

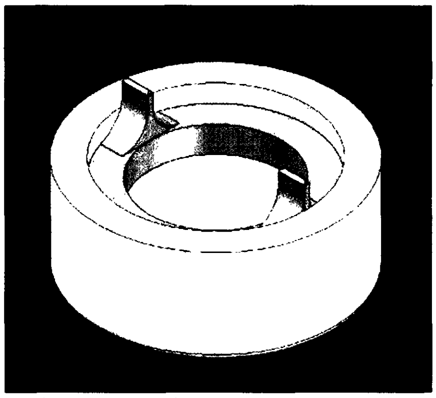


Figure 11: The submerged capsule design. The inner grey torus is threaded on the exterior and threads into the white outer cylinder. The entire cylinder is submerged in water. (Image created in SolidEdge [15].)

The capsule portion was made up of an inner and outer portion that screwed together, much like the top of a medicine bottle. Contained inside the capsule were the two pieces of mesh to contain the worm and to provide the electrical stimulation. The mesh pieces were connected to wires and separated by an acrylic ring. This capsule could be closed quickly and easily by hand and then dropped into the water. The mesh would allow the air to pass out of the chamber, eliminating air bubbles. Also, the seals on this design were limited to those in water chamber that the capsule was dropped into. Such water containers are an existing technology, and therefore the leakage issues would be eliminated.

A few examples of the submerged capsule design were built and tested [Figure 12].

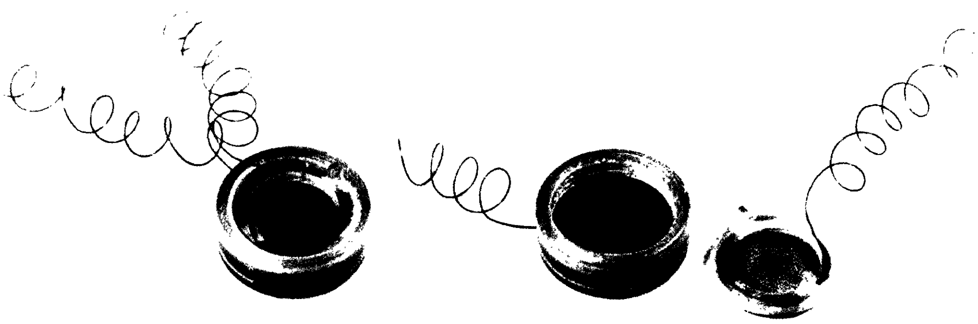


Figure 12: The submerged capsule, not in water. Capsule on the left is assembled, capsule on right shows the top portion removed.

The capsule was manufactured using the SLA. The mesh disks were epoxied to the different sides of the threaded portions so that the user would not have to position them by hand. A wire was attached to each mesh disk so that it could be electrified. An acrylic ring was epoxied to the bottom of the capsule so that when the inner threaded piece was screwed into place, the mesh would be a specified distance apart.

The capsules were tested. There was no leakage from the acrylic containers that the capsules were placed in. The air did not escape from between the mesh disks as easily as hoped, but the air bubbles were eliminated when the capsule was tightened underwater. Future work will focus on making the mesh less hydrophobic. When planaria were placed inside the capsules, they tended to stay on the sides of the chamber, on the acrylic ring as opposed to the mesh disks. However, they did move around and did not show signs of lethargy or stress when left in the chamber for 2 weeks. This design was deemed successful enough for use when attempting to condition worms.

3.2 Electronics Design

The electronics portion of the training environment design had one goal: to provide stimulation to the planaria in the forms of light and/or shock. This was met by a prototype system of Agilent Data Acquisition System [22] and various outputs from the system.

3.2.1 The Resistivity of Poland Spring Water

Due to the differences in geometry between the cylindrical chamber design and previous trough technology, the electrical properties of Poland Spring Water had to be analyzed in order to duplicate the electrical stimulation to the planaria. Poland Spring Water (PSW) is a spring water, so it contains numerous impurities such as salts and minerals. Therefore, it was necessary to determine the resistivity through experimentation. Based on a new high-accuracy method that did not require cell-factor calibration [23], the resistivity of PSW was tested. A platinum coaxial-cylinders probe was constructed, supported and separated by a cylinder that was fabricated in the SLA. The central probe was a Pt wire of 0.79 mm diameter, and the outer cylinder was a piece of Pt foil that had been rolled into a cylinder and soldered so that it was electrically continuous. The foil was 0.25 mm thick and when rolled into a cylinder, it had an inner diameter of 8 mm. Pt wire (300 μm diameter) was soldered to the top of the center probe and held to the outer cylinder using silicon tubing. The entire probe was mounted on a vertical linear stage so that it could be moved in the vertical direction by specific amounts. A beaker of the liquid in question was then placed underneath. The probe was lowered into the solution [Figure 13] and measurements taken using an HP4194A Impedance/Gain-Phase Analyzer.

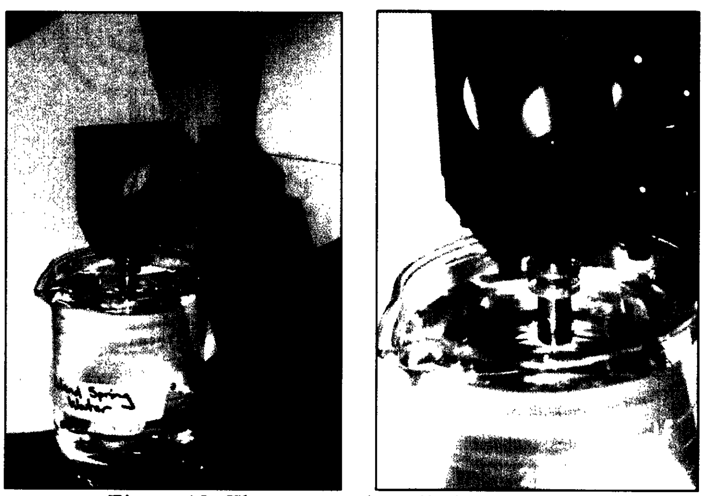


Figure 13: The concentric-cylinders Pt probe.

At a given depth, gain, frequency, and phase data was taken over a large range of frequencies. The probe was then lowered by one millimeter, and the measurement repeated. This was performed at eight or more different immersion depths for each solution. From this data, Bode plots and a Nyquist plot were produced for each depth [Figure 14] using Matlab [24] (code found in Appendix A).

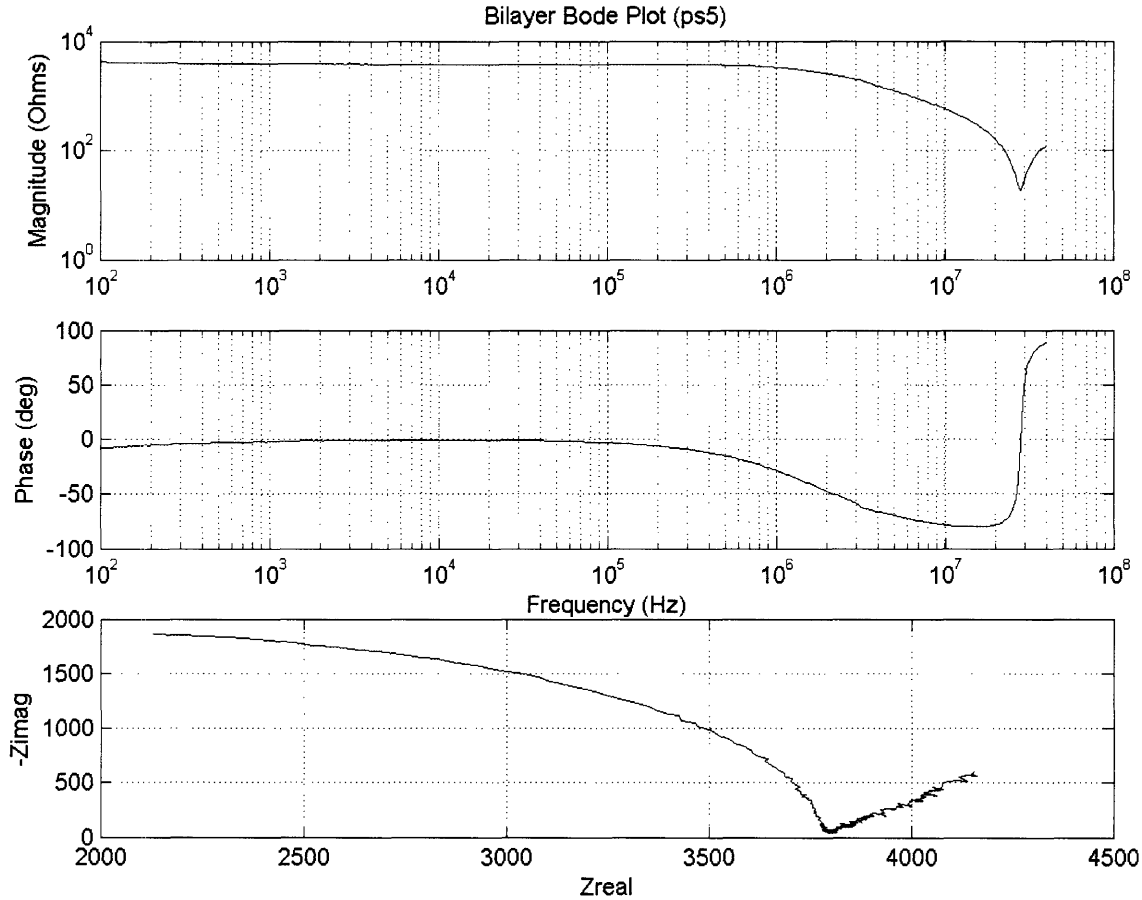


Figure 14: Bode plots and Nyquist plot for Poland Spring Water at immersion level 5.

The value of the real portion of the impedance at the minimum value of the imaginary part of the impedance was taken as the purely resistive part of the impedance in the solution. (A detailed explanation of this logic can be found in Reference [23]). By plotting these real portions of the impedance at each depth, a relationship can be found between the resistivity and the depth [Figure 15].

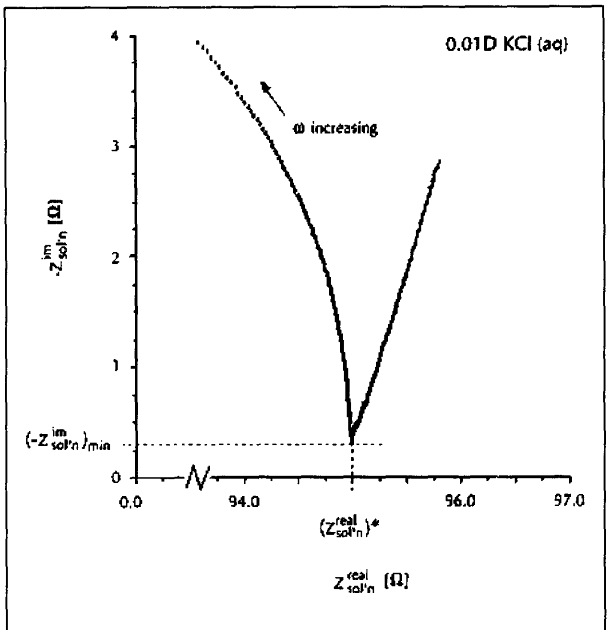


Figure 2-3: Typical Nyquist plot (0.01D KCl (aq) solution)

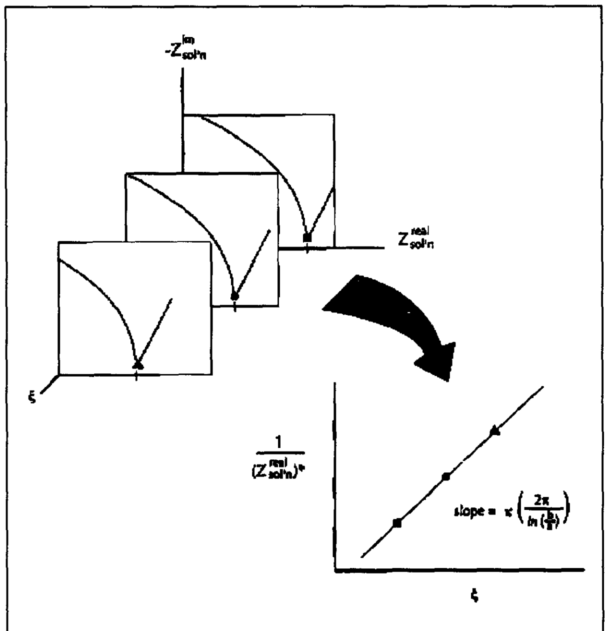


Figure 2-7: Graphical representation of how multiple $(Z^{real}_{sol'n})^*$ yield κ

Figure 15: Graphical Representation of how multiple values of the purely resistive measurement at different depths can produce a graph where the resistivity of the solution is simply a geometric factor divided by the slope of the line. (Plots taken from [23]).

The critical points in the Nyquist Plots were found using a Matlab program written for this application [Appendix A]. The points were then plotted in Excel [25] and the slope of the linear trend line was used to determine the resistivity [Figure 16]. Note the high R-squared value.

Results from Nyquist Plots for Poland Springs Water

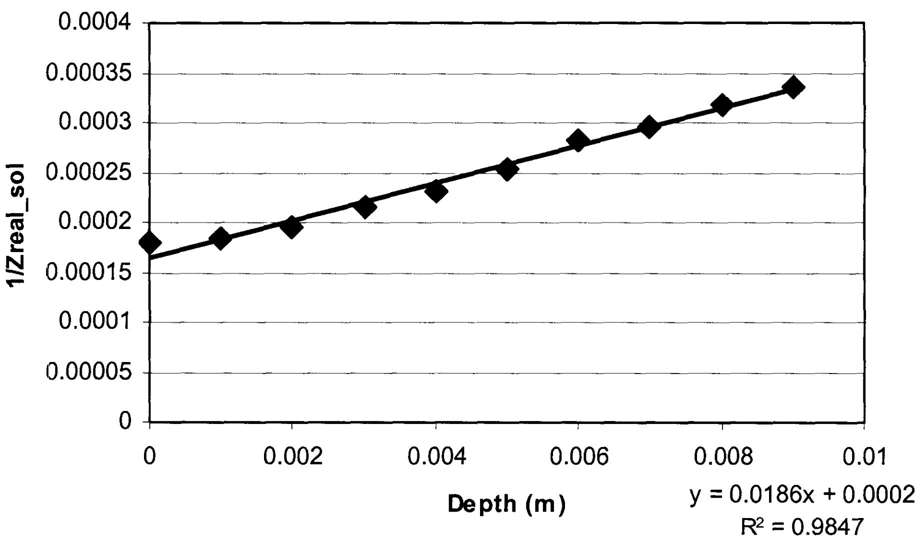


Figure 16: Excel chart of the critical points from the Nyquist plots of PSW. A linear trendline was fit using the least-squares method, where x is 1/Zreal_sol and y is depth in meters. Variance accounted for (R^2) by the linear model is also shown.

Initially, only PSW and Millipore water were tested. The Millipore water was meant to be a control for the experiment because its resistivity is very well-defined. However, after many trials it became clear that its resistivity was too high, beyond the limits of this method of measurement. Measurements were done of potassium chloride solutions, the same concentrations tested in the original paper describing this method. These tests verified that the probe and methods were reasonably accurate (within 20% of accepted values [26], despite not performing the experiments in a nitrogen environment). Therefore, other solutions useful to lab members were tested as well, with the results summarized in Table 1.

Table 1: Resistivity and Conductivity of Various Solutions, Measured Using the Concentric Probe Method

	Resistivity Ω·m	*Conductivity* $(\Omega\cdot m)^{-1}$
Sea Water	0.33	3.0
PC conc. 1	7.8	0.13
PC conc. 2	4.4	0.23
BF4-	2.7	0.38
PF6-	6.0	0.17
PC+0.6M TEAP	1.65	0.61
AN+0.1M TEAP	0.95	1.1
Poland Spring	138	0.01

This experiment increased the understanding of the stimulation that the planaria were receiving during conditioning and allowed for the prototype to approximate the normal conditioning experience.

3.2.2 System Lighting and Observation

The training environment required light as a stimulation, and a way to visualize the planaria while they were in the device. These two requirements led to the utilization of infrared LEDs (light-emitting diodes) with peak spectra of 940 nm in combination with a digital imagining device that can detect infrared light waves. Infraded light is outside of the spectral range of the planarian photoreceptors [27]. For prototyping purposes, an iBOT camera from Orange Micro, Inc [21] was used due to its low price and ability to capture at high frame-rates as a result of its USB2 connection.

3.2.3 Agilent Prototyping Equipment

An Agilent Data Acquisition / Switch unit (34970A) with a 20-channel multiplexer card (HP 34901A) was used to provide the electrical stimulus. This was an easy way to test the system without the complications of building switches or circuits. Also, this allowed easy integration with the computer program that was running the experiment due to the Agilent Toolkit software program.

3.3 Computer Programming

The computer program part of the training environment needed to run the experiment: the lights and electrical stimulation, while providing an intuitive graphical interface. This would make it simple for a researcher set up experiments and analyze the subsequent data. However, the program also had to automatically keep track of the data and make sure that the experiments were run in a way that had imbedded safety features for the planaria.

3.3.1 Visual Basic .NET Program

Two versions of the computer program were created using Visual Basic .NET [25]. The first was based on the idea that the worm would be stimulated based on its position in the chamber (Instrumental conditioning). Until the control loop was closed using a digital camera, the program would allow the user to choose when to stimulate the planaria. The program would also record the instances of stimulation and would graphically display the learning exhibited by the planaria using Microsoft Excel. An outline of this program sequence is shown in Figure 17 and the Visual Basic .NET code is given in Appendix B.

The second version of the computer program was created to train the planaria using Classical conditioning. This would stimulate them with light and then after a few seconds of light give them a shock. Their response during the light stimulation time would be recorded. An outline of this program sequence is shown in Figure 18 and the Visual Basic .NET code is give in Appendix C.

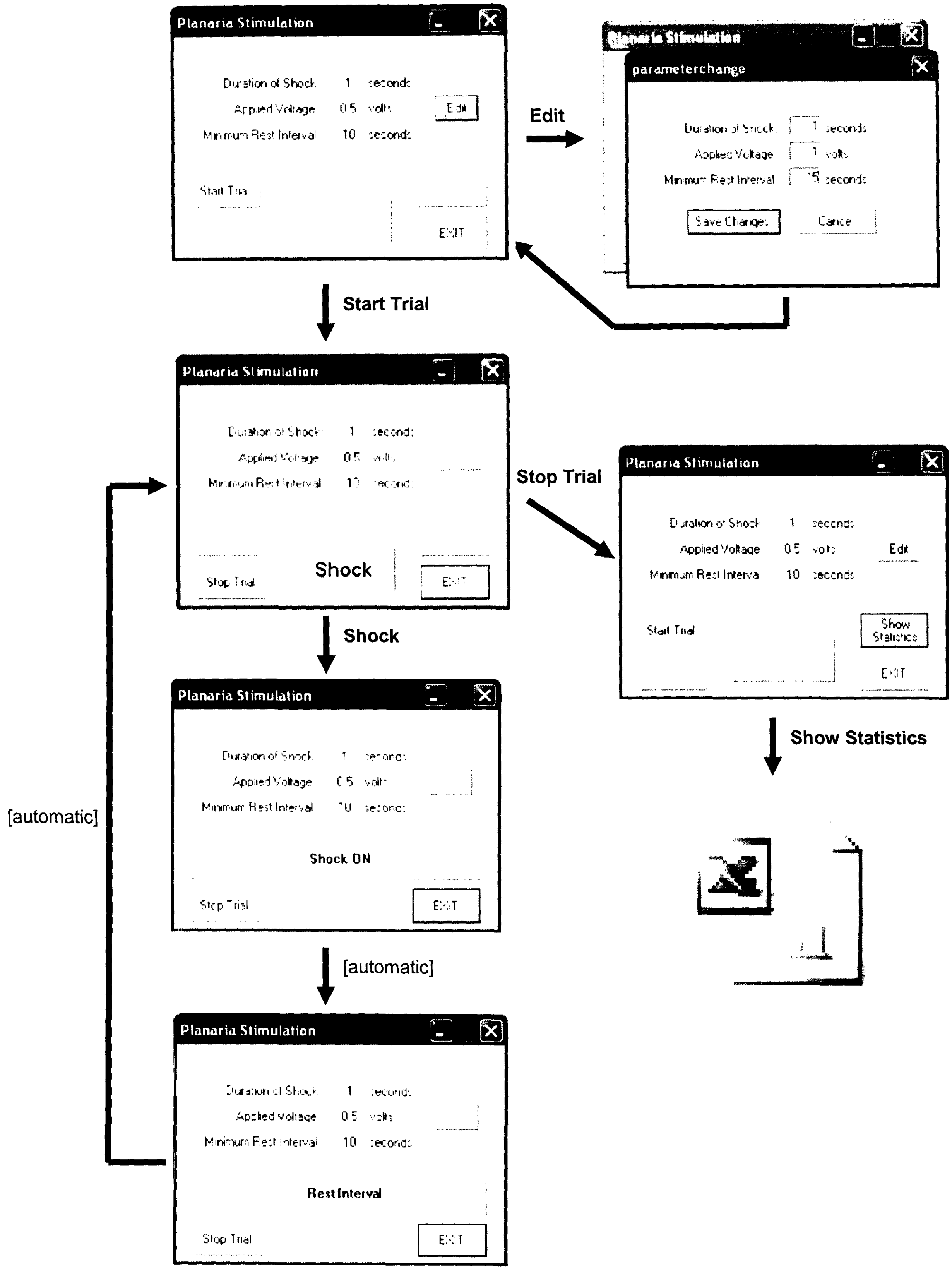


Figure 17: Program sequence for the conditioning of planaria based on their location in the chamber (Instrumental Conditioning).

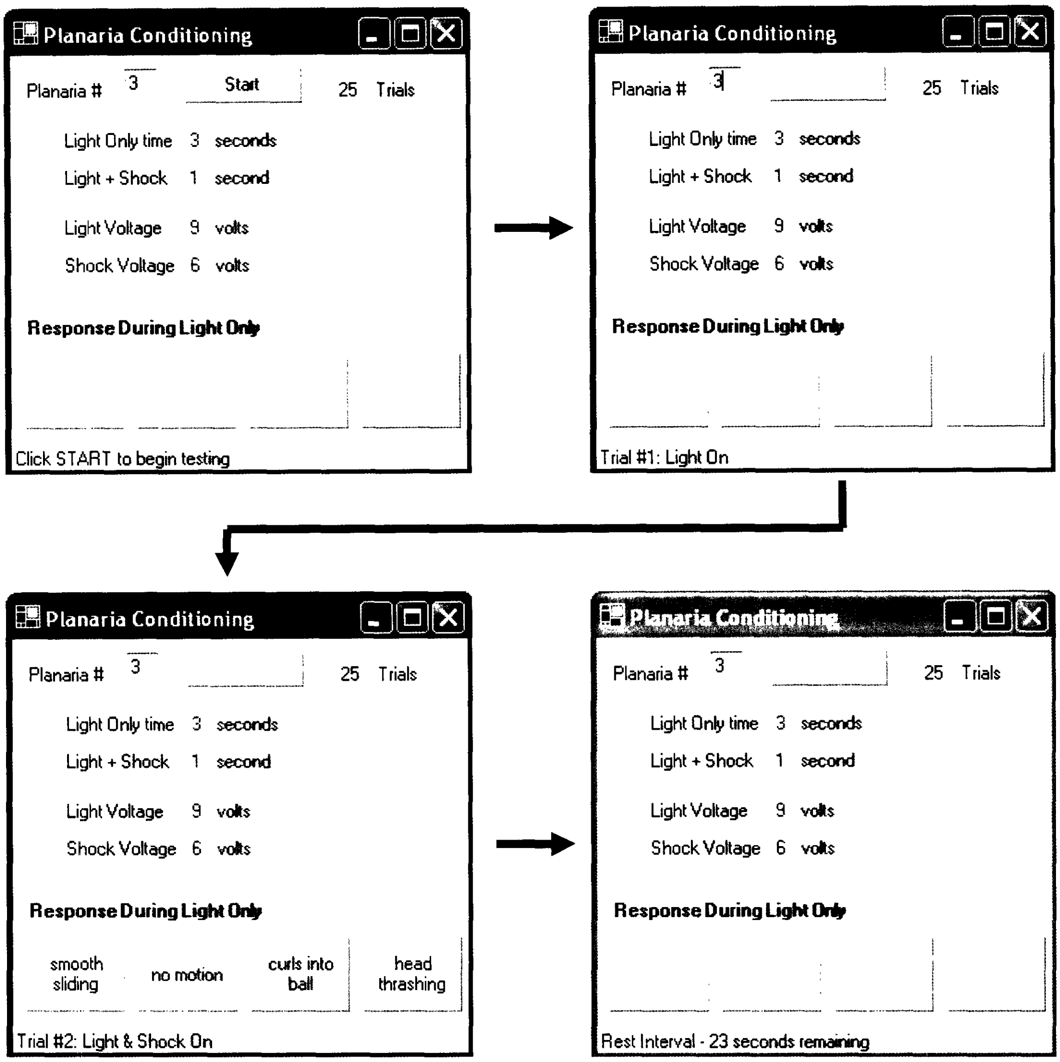
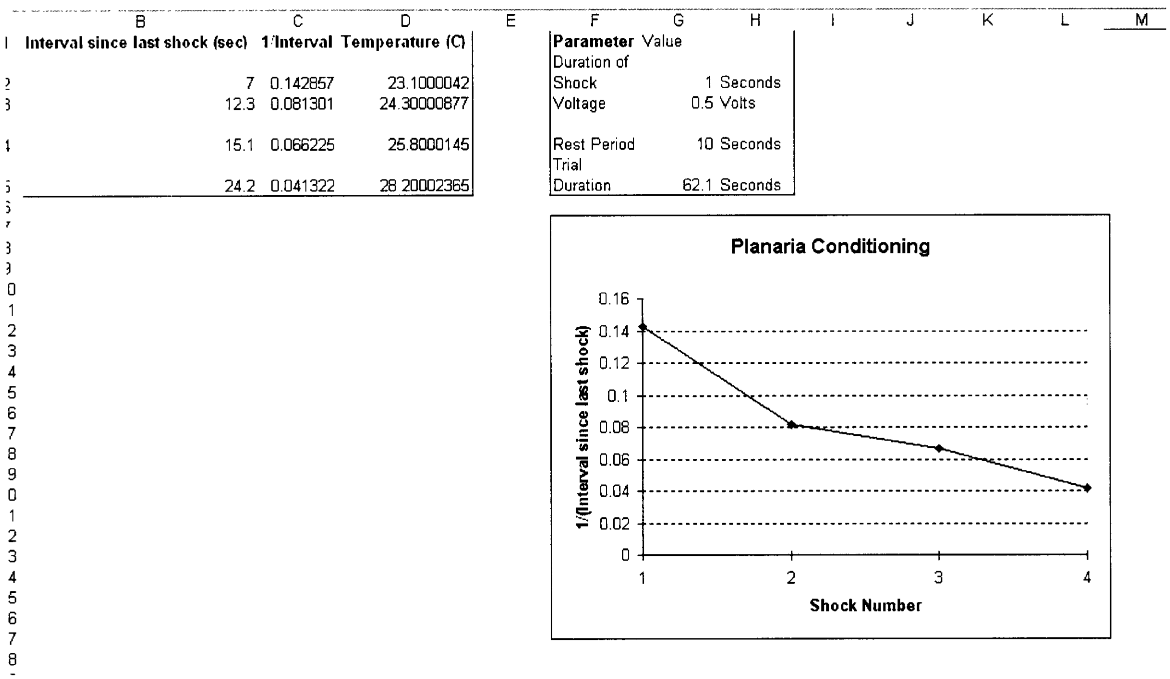


Figure 18: Program sequence for the conditioning of planaria using the Classical conditioning method.

3.3.2 Excel Data Recording

The data from the Visual Basic programs were automatically recorded into a Microsoft Excel spreadsheet. For the program that records Instrumental conditioning, the time between stimulation was recorded, as well as all of the parameters of voltage, duration, and minimum rest interval. When the experiment ended, the program graphed the results in the format of the inverse of the time between shocks versus the number of the shock (Figure 19). Therefore, if the planaria was learning successfully, the time between shocks would increase (the planaria would not need as frequent stimulation in order to stay in the correct part of the chamber) and so the graph of the inverse of the time between shocks would approach zero.

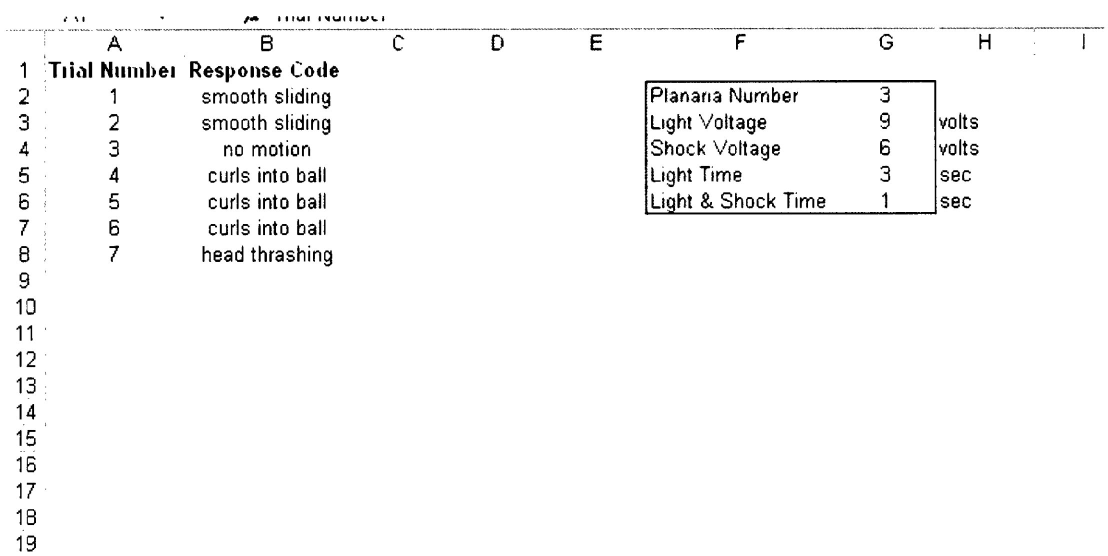


Interval since last shock (sec)	1/Interval	Temperature (C)
7	0.142857	23.1000042
12.3	0.081301	24.30000877
15.1	0.066225	25.8000145
24.2	0.041322	28.20002365

Parameter	Value
Duration of Shock	1 Seconds
Voltage	0.5 Volts
Rest Period	10 Seconds
Trial Duration	62.1 Seconds

Figure 19: Screen capture of an Excel worksheet showing the parameters recorded automatically by the Visual Basic .NET program for Instrumental conditioning.

The program for Classical conditioning recorded the response of the planaria during the light portion of the trail as well as the operating conditions, such as voltage and duration of each of the stimuli (Figure 20).



Trial Number	Response Code
1	smooth sliding
2	smooth sliding
3	no motion
4	curls into ball
5	curls into ball
6	curls into ball
7	head thrashing

Planaria Number	3	
Light Voltage	9	volts
Shock Voltage	6	volts
Light Time	3	sec
Light & Shock Time	1	sec

Figure 20: Screen capture of an Excel worksheet showing the parameters recorded automatically by the Visual Basic .NET program for Classical conditioning.

4 Conditioning of Planaria Using the Planaria Training Environment

In order to test the training environment, experiments were run that attempted to condition planaria using classical conditioning methods. These methods have been experimentally verified using the tradition trough set-up [28].

4.1 Experimental Set-Up

Ten chambers were machined and assembled as described in Section 3.1.5. A planaria was placed in each chamber and each chamber was then submerged in Poland Spring Water, each chamber in an individual plastic box [Figure 21].

Figure 21: A training environment chamber inside a plastic container, ready for testing.

4.2 Procedure for Conditioning

The plastic box containing the planaria in the training chamber was placed on top of a clear base that was illuminated from beneath with infrared LEDs. The chamber was imaged from above using the iBOT camera. The wires from the chamber were connected to the Agilent Data Acquisition / Switch Unit and the Classical conditioning Visual Basic .NET program was run. Data was recorded in Microsoft Excel spreadsheets.

4.3 Results

The planaria did not respond well in the training chambers. They much preferred to stay on the sides of the chamber, which was not useful in this classical conditioning experiment because they were not visible on the sides of the chamber, and therefore their response to the light was not recorded.

Also, the planaria did not appear to live as healthily as the test subject who lived in the chamber for two weeks and was not subjected to any experimentation while in the chamber. This might have been a result of the increased stress that they experienced during the training procedure.

Another issue that made the training environment not very amenable to classical conditioning is that it isolated the planaria in a manner that did not allow it to be prodded in any way during the experimentation. Often researchers have to "nudge" the planaria when they stop moving during the experiment [29] in order to keep them from balling up and not moving in any manner. This was not able to be performed with the planaria sealed inside the capsules.

All of these issues lead to the early termination of the experiments.

4.4 Discussion of the Experiment

The experiment did not accomplish its stated goals in that it did not successfully condition planaria, but this did not mean that it was a failure. This apparatus was not designed with classical conditioning in mind. It was originally meant to be used for instrumental conditioning, based on the location of the planaria in the chamber. However, it was decided that the classical conditioning experiment should be tried first, since it would more directly correlate to other experiments that had already been performed that proved that planaria could learn to associate light with shock. No experiments have been done to determine if planaria are capable of instrumental learning based on location. It was hoped that if the planaria were successfully conditioned using classical methods in the new training environment that it would signify the "approval" of the design by the planaria, and then experiments involving instrumental conditioning would begin.

The experiment might not have worked for several reasons. Most likely the planaria were not able to have enough trials because the planaria were not able to be tested when they were on the sides of the chamber. This could be solved by either decreasing the height of the chamber so there isn't as much wall space for them to stay on, or by angling the walls so they are never vertically-oriented and therefore the camera would be able to see them at all times.

Another possibility is that the planaria were not comfortable enough in the chamber environment to be able to be conditioned. However, the validity of this conjecture cannot be determined because of the inability to be able to run the testing procedure as described because of the planarias' preference for staying on the sides of the chambers. It is interesting to note that the planaria were not stimulated while they were on the sides of the chamber. This is almost like running an instrumental conditioning experiment where the planaria are not shocked if they stay in the appropriate part of the chamber. However, this was not proven either since the naïve planaria prefer the sides of the chamber to begin with.

5 Conclusion and Future Directions

The design of the planaria training environment opens new possibilities for autonomously-conditioning numerous planaria in parallel. This would be an exciting method of instrumenting at the level of the organism that would allow for an advanced pace of scientific research into areas such as behavior and memory.

The planaria training environment was not successful in its first experiment but for reasons about the planarian preference for the sides of the chamber that had not been foreseen. This does not mean that the training environment does not have potential to condition planaria. Further experimentation must be performed to see if the planaria can be conditioned classically or instrumentally using a modified version of the device which would allow them to be viewed even when they are on the side of the chamber.

Also, future work can be done to enhance the Visual Basic .NET programs. It would be useful to record temperature data to ensure that the water temperature does not go outside of the planaria's comfort zone.

The next step is to attempt to put many chambers in parallel when conditioning the planaria. Many new challenges may be faced when this expansion of the current design is attempted, such as water flow issues or integration of automatic computer feedback based on visual analysis. However, I am confident that the planaria training environment device will be successful in the future, and the fields of behavioral and memory research will be improved because of it.

Bibliography

[1] Andrews, B. J., Turchi, J. J., (2004). Development of a high-throughput screen for inhibitors of replication protein A and its role in nucleotide excision repair. Molecular Cancer Therpeutics: 4, 385-391.

[2] Peekhaus, N. T., Ferrer, M., et al, (2003). A bet-Lactamase-Dependant Gal4-Estrogen Receptor beta Transactivation Assay for the Ultra-High Throughput Screening of Estrogen Receptor beta Agonists in a 3,456-Well Format. Assay and Drug Development Technologies: 6, 789-800.

[3] Best, J. B., (1963). Protopsychology. Scientific American: 208, 55-62.

[4] Levin, M. http://www.drmichaellevin.org, http://www.forsyth.org/re/re_i_levin.htm.

[5] Thompson, R., McConnell, J., (1955). Classical Conditioning in the Planarian, Dugesia Dorotocephala. Journal of Comparative and Physiological Psychology: 48, 65-68.

[6] Best, J. B., Rubinstein, I., (1962). Maze Learning and Associated Behavior in Planaria. Journal of Comparative and Physiological Psychology: 55, 560-566.

[7] McConnell, J. V., Cornwall, P. R., Clay, M., (1960). An Apparatus for Conditioning Planaria. American Journal of Psychology: 73, 618-622.

[8] Gray, P. *Psychology*, 2nd Edition. New York: Worth, 1994.

[9] Ward's Natural Science Establishment, Working with Planaria, <www.wardsci.com>, 2002.

[10] Dasheiff, B. D., Dasheiff, R. M., (2002). Photonegative Response in Brown Planaria (Dugesia tigrina) Following Regeneration. Ecotoxicology and Environmental Safety: 53, 196-199.

[11] Hartry, Keith-Lee, Morton, (1964). Planaria: Memory Transfer through Cannibalism Reexamined. Science 146, 274-275.

[12] Mitman, G., Fausto-Sterling, A., "Whatever Happened to Planaria? C. M. Child and the Physiology of Inheritance." The Right Tools for the Job, Clarke, A. E., Fujimura, J., H., eds. Princeton University Press , Princeton, NJ: 1992, pp. 172-197.

[13] Kitamura, Y., Kakimura, J., Taniguchi, T., (1998). Protective Effect of Talipexole on MPTP-Treated Planarian, a Unique Parkinsonian Worm Model. Japan Journal of Pharmacology: 78, 23-29.

[14] Schaeffer, D. J., (1993). Planarians as a Model System for In Vivo Tumerogenesis Studies. Ecotoxicology and Environmental Safety: 25, 1-18.

[15] UGS PLM Solutions, Plano, Texas, USA, http://www.solid-edge.com.

[16] HAAS Automation, Inc., Oxnard, California, USA, http://www.haascnc.com

[17] Charmilles Technologies SA, Geneva, Switzerland, http://www.charmilles.com.

[18] Epilog, Golden, Colorado, USA, http://www.epiloglaser.com.

[19] 3D Systems, Valencia, California, USA, http://3dsystems.com.

[20] Para Tech Coating, Inc., Aliso Viejo, California, USA, http://www.parylene.com.

[21] Orange Micro, Inc., Anaheim, California, USA, http://www.orangemicro.com.

[22] Agilent Technologies, Inc., Palo Alto, California, USA, http://www.agilent.com.

[23] Schiefelbein, S. L., Fried, N. A., Rhoads, K. G., Sadoway, D. R., (1998). A high-accuracy, calibration-free technique for measuring the electrical conductivity of liquids. Review of Scientific Instruments: 69, 3308-3313.

[24] The MathWorks, Natick, Massachusetts, USA, http://www.mathworks.com.

[25] Microsoft Corporation, Redmond, Washington, USA, http://www.microsoft.com.

[26] Wu, Y. C., Koch, W. F., Hamer, W. J., (1987). Review of Electrolytic Conductance Standards. Journal of Solution Chemistry: 16, 985-997.

[27] Brown, H. M., Ogden, T. E., (1968). Spectral Sensitivity of the Planarian Ocellus. Journal of General Physiology: 51, 255-260.

[28] Block, R. A., McConnell, J. V., (1967). Classically Conditioned Descrimintaion in the Planarian, Dugesia dorotocephala. Nature 215, 1465-1466.

[29] Personal conversations with Debra Sorocco, researcher at the Forsyth Institute in Boston.

Additional Resources

Brown, H. M., Ogden, T. E., (1968). The Electrical Reponse of the Planarian Ocellus. Journal of General Physiology: 51, 237-253.

Gleitman, Fridlund, Reisberg. Phychology, 5th Edition. WWNorton Press: 1999. Library: BF121.058.

Jacobson, A. L., Horowitz, S. D., Fried, C., (1967). Classical Conditioning, Pseudoconditioning, or Sensitization in the Planarian. Journal of Comparative and Physiological Psychology: 64, 73-79.

McConnell, J. V., (1967). A manual of psychological experiments on planarians. Journal of Biological Psychology: 73, 618-622.

Appendix A: Matlab Code for Bode and Nyquist Plots

```
function Dbodefile2(name)

% name
%
% Dbodefile(name)
%
% loads in files (frequency, gain, phase) that have the specified names
% and plots them as bode plots.  Files must have the suffix .txt but
% DO NOT include that suffix in the argument passed to bodefile.
%
% The specified name of the file loaded are:
% freqNAME.txt          /frequency raw data
% gainNAME.txt          /gain raw data
% phaseNAME.txt         /phase raw data
%
% the loaded data are saved in a single text format (*.txt) file
% with the name savefile.txt
% In this file 3 colums are being displayed:
% FREQUENCY             GAIN            PHASE
%
freqname = ['freq', name];
gainname = ['gain', name];
phasename = ['phase',name];
savefilename = name;
eval(['load ', freqname, '.txt;']);
eval(['load ', gainname, '.txt;']);
eval(['load ', phasename, '.txt;']);
eval(['f = ',freqname, ';']);
eval(['g = ',gainname, ';']);
eval(['p = ',phasename, ';']);

% save original file names and the data:
varstr = [freqname, ',' ,gainname, ',' ,phasename];

mat = [f g p];
eval( [ 'save ',savefilename,'.txt mat -ASCII -DOUBLE -TABS' ] );


% Truncating low frequencies
freq=f(1:(length(f)-0));
gain=g(1:(length(g)-0));
phase=p(1:(length(p)-0));

% Find the magnitude value at the closest-to-zero phase
mini=min(abs(phase));
ResGain=gain(find(phase==mini))
ResGain=gain(find(phase==-mini))

% Gain plot
clf;
subplot(311);
loglog(f, g);
```

```
title(['Bilayer Bode Plot (', name, ')']);
ylabel('Magnitude (Ohms)');
grid on;

% phase plot
subplot(312);
semilogx(f, p);
xlabel('Frequency (Hz)');
ylabel('Phase (deg)');
grid on;

% Nyquist plot anaylsis
% Imaginary part of impedance is gain*sin(phase_angle)
% Real part of impedance is gain*cos(phase_angle)
p_rad=phase*2*pi/360;
Zreal=gain.*cos(p_rad);
Zimag=-gain.*sin(p_rad);

%Nyquist Plot
subplot(313);
plot(Zreal, Zimag);
xlabel('Zreal');
ylabel('-Zimag');
grid on;

Zreal_sol=Zreal(find(Zimag==min(Zimag)));
Zreal_sol
```

Appendix B: Visual Basic .NET Code for Instrumental Conditioning

This program was written by Dawn M. Wendell to allow a person observing the planaria through the camera images on their computer to stimulate the planaria in real-time with an electric shock delivered by the Agilent Data Acquisition / Switch Unit. The ability for temperature-sensing functionality was added but is not currently working. The data from the experiments are recorded in an Excel spreadsheet.

```
Imports Microsoft.Office.Core

Public Class Form1
  Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

  Public Sub New()
    MyBase.New()

    'This call is required by the Windows Form Designer.
    InitializeComponent()

    'Add any initialization after the InitializeComponent() call

  End Sub

  'Form overrides dispose to clean up the component list.
  Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
    If disposing Then
      If Not (components Is Nothing) Then
        components.Dispose()
      End If
    End If
    MyBase.Dispose(disposing)
  End Sub

  'Required by the Windows Form Designer
  Private components As System.ComponentModel.IContainer

  'NOTE: The following procedure is required by the Windows Form Designer
  'It can be modified using the Windows Form Designer.
  'Do not modify it using the code editor.
  Friend WithEvents btnSHOCK As System.Windows.Forms.Button
  Friend WithEvents lblSECONDS As System.Windows.Forms.Label
  Friend WithEvents tmrDURATION As System.Windows.Forms.Timer
  Friend WithEvents Label1 As System.Windows.Forms.Label
  Friend WithEvents lblVolts As System.Windows.Forms.Label
  Friend WithEvents Label2 As System.Windows.Forms.Label
  Friend WithEvents Label3 As System.Windows.Forms.Label
  Friend WithEvents tmrREST As System.Windows.Forms.Timer
  Friend WithEvents lblSHOCKON As System.Windows.Forms.Label
```

```
    Friend WithEvents btnCHANGE As System.Windows.Forms.Button
    Friend WithEvents lblVOLTAGE As System.Windows.Forms.Label
    Friend WithEvents lblREST As System.Windows.Forms.Label
    Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
    Friend WithEvents lblDUR As System.Windows.Forms.Label
    Friend WithEvents lblDURATION As System.Windows.Forms.Label
    Friend WithEvents btnEXIT As System.Windows.Forms.Button
    Friend WithEvents btnSTOP As System.Windows.Forms.Button
    Friend WithEvents btnSTAT As System.Windows.Forms.Button
    Friend WithEvents btnSTART As System.Windows.Forms.Button
    Friend WithEvents tmrTEST As System.Windows.Forms.Timer
    Friend WithEvents stbDegrC As System.Windows.Forms.Label
    Friend WithEvents tmrDEGR As System.Windows.Forms.Timer
    Friend WithEvents tmrLIGHT As System.Windows.Forms.Timer

    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        Me.components = New System.ComponentModel.Container
        Me.btnSHOCK = New System.Windows.Forms.Button
        Me.lblDUR = New System.Windows.Forms.Label
        Me.lblSECONDS = New System.Windows.Forms.Label
        Me.tmrDURATION = New System.Windows.Forms.Timer(Me.components)
        Me.Label1 = New System.Windows.Forms.Label
        Me.lblVolts = New System.Windows.Forms.Label
        Me.Label2 = New System.Windows.Forms.Label
        Me.Label3 = New System.Windows.Forms.Label
        Me.tmrREST = New System.Windows.Forms.Timer(Me.components)
        Me.lblSHOCKON = New System.Windows.Forms.Label
        Me.btnCHANGE = New System.Windows.Forms.Button
        Me.lblDURATION = New System.Windows.Forms.Label
        Me.lblVOLTAGE = New System.Windows.Forms.Label
        Me.lblREST = New System.Windows.Forms.Label
        Me.GroupBox1 = New System.Windows.Forms.GroupBox
        Me.btnEXIT = New System.Windows.Forms.Button
        Me.btnSTOP = New System.Windows.Forms.Button
        Me.btnSTAT = New System.Windows.Forms.Button
        Me.btnSTART = New System.Windows.Forms.Button
        Me.tmrTEST = New System.Windows.Forms.Timer(Me.components)
        Me.stbDegrC = New System.Windows.Forms.Label
        Me.tmrDEGR = New System.Windows.Forms.Timer(Me.components)
        Me.tmrLIGHT = New System.Windows.Forms.Timer(Me.components)
        Me.SuspendLayout()
        '
        'btnSHOCK
        '
        Me.btnSHOCK.Font = New System.Drawing.Font("Microsoft Sans Serif", 14.25!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.btnSHOCK.Location = New System.Drawing.Point(104, 152)
        Me.btnSHOCK.Name = "btnSHOCK"
        Me.btnSHOCK.Size = New System.Drawing.Size(96, 40)
        Me.btnSHOCK.TabIndex = 1
        Me.btnSHOCK.Text = "Shock"
        '
        'lblDUR
        '
        Me.lblDUR.Location = New System.Drawing.Point(40, 32)
        Me.lblDUR.Name = "lblDUR"
```

```
Me.lblDUR.TabIndex = 2
Me.lblDUR.Text = "Duration of Shock:"
Me.lblDUR.TextAlign = System.Drawing.ContentAlignment.MiddleRight
'
'lblSECONDS
'
Me.lblSECONDS.Location = New System.Drawing.Point(176, 32)
Me.lblSECONDS.Name = "lblSECONDS"
Me.lblSECONDS.Size = New System.Drawing.Size(56, 23)
Me.lblSECONDS.TabIndex = 3
Me.lblSECONDS.Text = "seconds"
Me.lblSECONDS.TextAlign = System.Drawing.ContentAlignment.MiddleLeft
'
'tmrDURATION
'
Me.tmrDURATION.Interval = 500
'
'Label1
'
Me.Label1.Location = New System.Drawing.Point(40, 56)
Me.Label1.Name = "Label1"
Me.Label1.TabIndex = 6
Me.Label1.Text = "Applied Voltage:"
Me.Label1.TextAlign = System.Drawing.ContentAlignment.MiddleRight
'
'lblVolts
'
Me.lblVolts.BackColor = System.Drawing.SystemColors.Control
Me.lblVolts.Location = New System.Drawing.Point(176, 56)
Me.lblVolts.Name = "lblVolts"
Me.lblVolts.Size = New System.Drawing.Size(32, 24)
Me.lblVolts.TabIndex = 7
Me.lblVolts.Text = "volts"
Me.lblVolts.TextAlign = System.Drawing.ContentAlignment.MiddleLeft
'
'Label2
'
Me.Label2.Location = New System.Drawing.Point(16, 80)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(128, 23)
Me.Label2.TabIndex = 8
Me.Label2.Text = "Minimum Rest Interval:"
Me.Label2.TextAlign = System.Drawing.ContentAlignment.MiddleRight
'
'Label3
'
Me.Label3.Location = New System.Drawing.Point(176, 80)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(48, 23)
Me.Label3.TabIndex = 9
Me.Label3.Text = "seconds"
Me.Label3.TextAlign = System.Drawing.ContentAlignment.MiddleLeft
'
'tmrREST
'
Me.tmrREST.Interval = 500
```

```
'
'lblSHOCKON
'
Me.lblSHOCKON.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.lblSHOCKON.ForeColor = System.Drawing.Color.Red
Me.lblSHOCKON.Location = New System.Drawing.Point(104, 128)
Me.lblSHOCKON.Name = "lblSHOCKON"
Me.lblSHOCKON.TabIndex = 12
Me.lblSHOCKON.TextAlign = System.Drawing.ContentAlignment.MiddleCenter
'
'btnCHANGE
'
Me.btnCHANGE.Location = New System.Drawing.Point(240, 56)
Me.btnCHANGE.Name = "btnCHANGE"
Me.btnCHANGE.Size = New System.Drawing.Size(40, 24)
Me.btnCHANGE.TabIndex = 13
Me.btnCHANGE.Text = "Edit"
'
'lblDURATION
'
Me.lblDURATION.Location = New System.Drawing.Point(144, 32)
Me.lblDURATION.Name = "lblDURATION"
Me.lblDURATION.Size = New System.Drawing.Size(32, 24)
Me.lblDURATION.TabIndex = 14
Me.lblDURATION.Text = "1"
Me.lblDURATION.TextAlign = System.Drawing.ContentAlignment.MiddleCenter
'
'lblVOLTAGE
'
Me.lblVOLTAGE.Location = New System.Drawing.Point(144, 56)
Me.lblVOLTAGE.Name = "lblVOLTAGE"
Me.lblVOLTAGE.Size = New System.Drawing.Size(32, 23)
Me.lblVOLTAGE.TabIndex = 15
Me.lblVOLTAGE.Text = "0.5"
Me.lblVOLTAGE.TextAlign = System.Drawing.ContentAlignment.MiddleCenter
'
'lblREST
'
Me.lblREST.Location = New System.Drawing.Point(144, 80)
Me.lblREST.Name = "lblREST"
Me.lblREST.Size = New System.Drawing.Size(32, 24)
Me.lblREST.TabIndex = 16
Me.lblREST.Text = "10"
Me.lblREST.TextAlign = System.Drawing.ContentAlignment.MiddleCenter
'
'GroupBox1
'
Me.GroupBox1.Location = New System.Drawing.Point(8, 16)
Me.GroupBox1.Name = "GroupBox1"
Me.GroupBox1.Size = New System.Drawing.Size(288, 96)
Me.GroupBox1.TabIndex = 17
Me.GroupBox1.TabStop = False
'
'btnEXIT
'
```

```
Me.btnEXIT.Location = New System.Drawing.Point(224, 168)
Me.btnEXIT.Name = "btnEXIT"
Me.btnEXIT.Size = New System.Drawing.Size(64, 32)
Me.btnEXIT.TabIndex = 18
Me.btnEXIT.Text = "EXIT"
'
'btnSTOP
'
Me.btnSTOP.Location = New System.Drawing.Point(16, 168)
Me.btnSTOP.Name = "btnSTOP"
Me.btnSTOP.Size = New System.Drawing.Size(64, 32)
Me.btnSTOP.TabIndex = 19
Me.btnSTOP.Text = "Stop Trial"
'
'btnSTAT
'
Me.btnSTAT.Location = New System.Drawing.Point(224, 128)
Me.btnSTAT.Name = "btnSTAT"
Me.btnSTAT.Size = New System.Drawing.Size(64, 32)
Me.btnSTAT.TabIndex = 20
Me.btnSTAT.Text = "Show Statistics"
'
'btnSTART
'
Me.btnSTART.Location = New System.Drawing.Point(16, 128)
Me.btnSTART.Name = "btnSTART"
Me.btnSTART.Size = New System.Drawing.Size(64, 32)
Me.btnSTART.TabIndex = 21
Me.btnSTART.Text = "Start Trial"
'
'tmrTEST
'
'stbDegrC
'
Me.stbDegrC.Location = New System.Drawing.Point(0, 208)
Me.stbDegrC.Name = "stbDegrC"
Me.stbDegrC.Size = New System.Drawing.Size(304, 16)
Me.stbDegrC.TabIndex = 22
'
'tmrDEGR
'
Me.tmrDEGR.Interval = 50000
'
'tmrLIGHT
'
Me.tmrLIGHT.Interval = 30000
'
'Form1
'
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.ClientSize = New System.Drawing.Size(304, 224)
Me.Controls.Add(Me.stbDegrC)
Me.Controls.Add(Me.btnSTART)
Me.Controls.Add(Me.btnSTAT)
Me.Controls.Add(Me.btnSTOP)
Me.Controls.Add(Me.btnEXIT)
```

```
        Me.Controls.Add(Me.lblREST)
        Me.Controls.Add(Me.lblVOLTAGE)
        Me.Controls.Add(Me.lblDURATION)
        Me.Controls.Add(Me.btnCHANGE)
        Me.Controls.Add(Me.lblSHOCKON)
        Me.Controls.Add(Me.Label3)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.lblVolts)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.lblSECONDS)
        Me.Controls.Add(Me.lblDUR)
        Me.Controls.Add(Me.btnSHOCK)
        Me.Controls.Add(Me.GroupBox1)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
        Me.MaximizeBox = False
        Me.Name = "Form1"
        Me.Text = "Planaria Stimulation"
        Me.ResumeLayout(False)

    End Sub

#End Region

    'This program applies a voltage using a DAC from a Agilent 34970A Data
    'Aquisition/Switch Unit across two wires for a specified amount of time.
    'The program records data about the interval between shocks and outputs
    'that data to an Excel spreadsheet.

    'Adding temp sensing functionality 9/23/03 - need Agilent code
    'figured out how to manipulate Excel chart labels 11/12/03

    'Written by Dawn Wendell with help from Jeff Hoff 8/8/03 for use with the
    'Planaria Training Environment.
    'Revision 6

#Region "variables & declarations"

    Public DIGITAL As Agilent.TMFramework.InstrumentIO.DirectIO

    Public DURATION As Single = 1000
    Public VOLTAGE As Single = 0.5
    Public REST As Single = 10000
    Private EXP_ON As Boolean = False
    Public milliTIME As Integer = 0
    Public Shock_Num As Integer = 0
    Private LastTime As Integer = 0
    Private DegrC As Single = 22
    Public minDegrC As Single = 18
    Public maxDegrC As Single = 35
    Private DegrBuffer As Integer = 5
    Private DegrExtreme As Integer = DegrBuffer + 2
    Public MyParameterchange As parameterchange
    Public MyStat As Statistics

    Private AllowSetup As Boolean = True
    Dim oXL As Excel.Application
```

```
    Dim oWB As Excel.Workbook
    Dim oSheet As Excel.Worksheet
    Dim oRng As Excel.Range
    Dim oChart As Excel.Chart
    Dim oSeries As Excel.Series

    Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load

        tmrDEGR.Interval = 1000
        tmrDEGR.Enabled = True

        btnSHOCK.Enabled = False
        btnSTOP.Enabled = False
        btnSTAT.Enabled = False

    End Sub

    Friend Function saveit(ByVal dur As Single, ByVal rest As Single, ByVal volt As Single)
        Me.VOLTAGE = volt
        Me.DURATION = dur
        Me.REST = rest
    End Function

#End Region

#Region "Timers"

    Private Sub tmrDURATION_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
tmrDURATION.Tick

        'Turns shock off

        tmrDURATION.Enabled = False
        DIGITAL.WriteLine("SOUR:VOLT 0.0, (@204)")
        lblSHOCKON.Text = "Rest Interval"

        tmrREST.Enabled = True
        tmrREST.Interval = REST

    End Sub

    Private Sub tmrREST_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles tmrREST.Tick

        'Rest timer counter

        tmrREST.Enabled = False
        lblSHOCKON.Text = ""

        If btnSTOP.Enabled = True Then
            btnSHOCK.Enabled = True
        Else
            btnSHOCK.Enabled = False
        End If

    End Sub
```

```
    Private Sub tmrTEST_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles tmrTEST.Tick

        'counts up the clock
        milliTIME += 100   '100 milliseconds

    End Sub

    Private Sub tmrLIGHT_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles tmrLIGHT.Tick

        'keeps light on for 30 sec

    End Sub

#End Region

#Region "Buttons: Shock, Parameters, Statistics, Exit"

    Private Sub btnSHOCK_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
btnSHOCK.Click

        ' This code creates and initializes an instance of the DirectIO class
        ' and uses hard-coded values for driver settings.

        DIGITAL = New Agilent.TMFramework.InstrumentIO.DirectIO("GPIB0::9::INSTR", False, 2000000)
        DIGITAL.Timeout = 2000000  'timeout in milliseconds

        'Checks for paramters input and rest interval

        If EXP_ON = False Then
            MessageBox.Show("Press the Start Trial button to start a new trial", "Error", _
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

        ElseIf tmrREST.Enabled = True Then
            lblSHOCKON.Text = "Rest Interval"

            'Turns on Shock

        Else
            DIGITAL.WriteLine("SOUR:VOLT " & lblVOLTAGE.Text & ", (@204)")
            lblSHOCKON.Text = "Shock ON"
            tmrDURATION.Enabled = True
            tmrDURATION.Interval = DURATION
            Shock_Num += 1
            oSheet.Cells(Shock_Num + 1, 1).value = Shock_Num
            oSheet.Cells(Shock_Num + 1, 2).value = (milliTIME - LastTime) / 1000
            oSheet.Cells(Shock_Num + 1, 3).value = 1 / ((milliTIME - LastTime) / 1000)
            oSheet.Cells(Shock_Num + 1, 4).value = DegrC
            If DegrC < minDegrC And DegrC > (minDegrC - DegrBuffer) Then
                oSheet.Cells(Shock_Num + 1, 4).interior.colorindex = 36
            ElseIf DegrC > maxDegrC And DegrC < (maxDegrC + DegrBuffer) Then
                oSheet.Cells(Shock_Num + 1, 4).interior.colorindex = 36
            ElseIf DegrC >= (maxDegrC + DegrBuffer) Then
                oSheet.Cells(Shock_Num + 1, 4).interior.colorindex = 45
            ElseIf DegrC <= (minDegrC - DegrBuffer) Then
                oSheet.Cells(Shock_Num + 1, 4).interior.colorindex = 45
            End If
```

```
        LastTime = milliTIME

        btnSHOCK.Enabled = False

        End If

    End Sub

    Private Sub btnCHANGE_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
btnCHANGE.Click

      If AllowSetup = False Then
        MessageBox.Show("Unable to change parameters during a trial. Please stop trial before editing parameters.",
_
        "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

      Else

        If IsNothing(MyParameterchange) = False Then
          'We have to check if its Nothing, If it is and you try to "Show()" it you will
          'get an error
          If MyParameterchange.IsDisposed = False Then
            'The form can be "Something" (not nothing) but be disposed. Calling the show
            'method on a disposed form will cause an error.
            'We dont do anything here because the form is not nothing, and its not
            'disposed so it must be open
            MyParameterchange.Show()
          Else 'The form isnt nothing but it is disposed.
            MyParameterchange = New parameterchange
            MyParameterchange.Show()
          End If
        Else 'The form is nothing
          MyParameterchange = New parameterchange
          MyParameterchange.Show()
        End If
        MyParameterchange.GiveAccess(Me)
        'MyParameterchange.Show()

      End If

    End Sub

    Private Sub btnEXIT_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btnEXIT.Click

      'Exits the program
      oRng = Nothing
      oSheet = Nothing
      oWB = Nothing
      oXL = Nothing
      Me.Close()

    End Sub

    Private Sub btnSTAT_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btnSTAT.Click
```

```
        If AllowSetup = True Then
            'Shows the excel spreadsheet
            oXL.WindowState = Excel.XlWindowState.xlNormal
        Else
            MessageBox.Show("Please complete a trial before viewing statistics", "Error", MessageBoxButtons.OK, _
                MessageBoxIcon.Exclamation)
        End If

    End Sub

#End Region

#Region "Start Button and Excel Initialization"

    Private Sub btnSTART_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
btnSTART.Click

        btnCHANGE.Enabled = False
        btnSTAT.Enabled = False
        btnSTART.Enabled = False
        btnSHOCK.Enabled = True
        btnSTOP.Enabled = True

        'Start Excel and get Application object
        oXL = CreateObject("Excel.Application")
        oXL.WindowState = Excel.XlWindowState.xlMinimized
        oXL.Visible = True

        'Get a new workbook.
        oWB = oXL.Workbooks.Add
        oSheet = oWB.ActiveSheet

        'Sets Column Titles
        oSheet.Cells(1, 1).value = "Shock Number"
        oSheet.Cells(1, 2).value = "Interval since last shock (sec)"
        oSheet.Cells(1, 3).value = "1/Interval"
        oSheet.Cells(1, 4).value = "Temperature (C)"
        oSheet.Cells(1, 6).value = "Parameter"
        oSheet.Cells(1, 7).value = "Value"

        'Format titles
        With oSheet.Range("A1", "F1")
            .Font.Bold = True
            .VerticalAlignment = Excel.XlVAlign.xlVAlignCenter
        End With
        oRng = oSheet.Range("A1", "F50")
        oRng.EntireColumn.AutoFit()
        oRng.WrapText = True

        'Records Settings
        oSheet.Cells(2, 6).value = "Duration of Shock"
        oSheet.Cells(3, 6).value = "Voltage"
        oSheet.Cells(4, 6).value = "Rest Period"
        oSheet.Cells(5, 6).value = "Trial Duration"
        oSheet.Cells(2, 7).value = DURATION / 1000
        oSheet.Cells(3, 7).value = VOLTAGE
```

```
        oSheet.Cells(4, 7).value = REST / 1000
        oSheet.Cells(2, 8).value = "Seconds"
        oSheet.Cells(3, 8).value = "Volts"
        oSheet.Cells(4, 8).value = "Seconds"
        oSheet.Cells(5, 8).Value = "Seconds"

        oRng = oSheet.Range("F1", "H5")
        With oRng.Borders(Excel.XlBordersIndex.xlEdgeBottom)
            .LineStyle = Excel.XlLineStyle.xlContinuous
            .Weight = Excel.XlBorderWeight.xlThin
        End With
        With oRng.Borders(Excel.XlBordersIndex.xlEdgeTop)
            .LineStyle = Excel.XlLineStyle.xlContinuous
            .Weight = Excel.XlBorderWeight.xlThin
        End With
        With oRng.Borders(Excel.XlBordersIndex.xlEdgeLeft)
            .LineStyle = Excel.XlLineStyle.xlContinuous
            .Weight = Excel.XlBorderWeight.xlThin
        End With
        With oRng.Borders(Excel.XlBordersIndex.xlEdgeRight)
            .LineStyle = Excel.XlLineStyle.xlContinuous
            .Weight = Excel.XlBorderWeight.xlThin
        End With

        'Restarts counters for new experiment
        EXP_ON = True
        milliTIME = 0
        LastTime = 0
        Shock_Num = 0
        tmrTEST.Enabled = True
        tmrTEST.Interval = 100   'time in milliseconds

    End Sub

#End Region

#Region "Stop Button and Excel finishing (Graphs Results)"

    Private Sub btnSTOP_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btnSTOP.Click

        'Stops clock and experiment
        tmrTEST.Enabled = False
        EXP_ON = False
        lblSHOCKON.Text = ""
        btnCHANGE.Enabled = True
        btnSTART.Enabled = True
        btnSTAT.Enabled = True
        btnSHOCK.Enabled = False
        btnSTOP.Enabled = False

        oSheet.Cells(5, 7).Value = milliTIME / 1000

        oRng = oSheet.Range("A1", "D" & Shock_Num + 1)
        With oRng.Borders(Excel.XlBordersIndex.xlEdgeBottom)
            .LineStyle = Excel.XlLineStyle.xlContinuous
            .Weight = Excel.XlBorderWeight.xlThin
```

```
        End With
        With oRng.Borders(Excel.XlBordersIndex.xlEdgeTop)
            .LineStyle = Excel.XlLineStyle.xlContinuous
            .Weight = Excel.XlBorderWeight.xlThin
        End With
        With oRng.Borders(Excel.XlBordersIndex.xlEdgeLeft)
            .LineStyle = Excel.XlLineStyle.xlContinuous
            .Weight = Excel.XlBorderWeight.xlThin
        End With
        With oRng.Borders(Excel.XlBordersIndex.xlEdgeRight)
            .LineStyle = Excel.XlLineStyle.xlContinuous
            .Weight = Excel.XlBorderWeight.xlThin
        End With

        'Adds a Chart of the 1/Interval data
        oRng = oSheet.Range("C2", "C" & Shock_Num + 1)
        oChart = oSheet.Parent.Charts.Add
        With oChart
            .ChartWizard(oRng, Excel.XlChartType.xlLine, , Excel.XlRowCol.xlColumns)
            oSeries = .SeriesCollection(1)
            oSeries.XValues = oSheet.Range("A2", "A" & Shock_Num + 1)

            .Axes(Excel.XlAxisGroup.xlPrimary).HasTitle = True
            .Axes(Excel.XlAxisType.xlCategory).AxisTitle.Characters.Text = "Shock Number"
            .Axes(Excel.XlAxisGroup.xlSecondary).HasTitle = True
            .Axes(Excel.XlAxisType.xlValue).AxisTitle.Characters.Text = "1/(Interval since last shock)"
            .HasTitle = True
            .ChartTitle.Text = "Planaria Conditioning"
            .HasLegend = False
            .Location(Excel.XlChartLocation.xlLocationAsObject, oSheet.Name)

        End With

        'oChart.ChartArea.AxisX.Title.Text = "Sales Report"

        'Move the graph away from the data
        With oSheet.Shapes.Item("Chart 1")
            .Top = oSheet.Rows(7).Top
            .Left = oSheet.Columns(6).Left
        End With

    End Sub

#End Region

#Region "Measure Temperature"

    Private Sub tmrDEGR_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles tmrDEGR.Tick

        'For each tick of the Degrees timer, a measurement of the temperature is
        'taken from the Agilent Data Aquisition hardware and the temperature is
        'displayed on the status label.

        ' This code creates and initializes an instance of the DirectIO class
        ' and uses hard-coded values for driver settings.
        ' For applied voltage
```

```
        DIGITAL = New Agilent.TMFramework.InstrumentIO.DirectIO("GPIB0::9::INSTR", False, 2000000)
        DIGITAL.Timeout = 2000000  'timeout in milliseconds

        'need to edit to be temp sensing variable
        'DIGITAL = New Agilent.TMFramework.InstrumentIO.DirectIO("GPIB0::9::INSTR", False, 2000000)
        'DIGITAL.Timeout = 2000000  'timeout in milliseconds
        'MONITER TEMP
        '       DIGITAL.WriteLine("MEAS:TEMP? RTD,85,1,0.1,(@101)") 'GIVES ERROR: Query Interrupted
        'DIGITAL.ReadNumberAsSingle(@101)
        'NEEDS TO BE BOOLEAN: DegrC = DIGITAL.Read("MEAS:TEMP? RTD,85,1,0.1,(@101)")
        If DegrC < minDegrC And DegrC > (minDegrC - DegrExtreme) Then
            stbDegrC.ForeColor = System.Drawing.Color.Red
            stbDegrC.Text = "Low Temp. Warning: " & DegrC & " degrees Celcius"
        ElseIf DegrC > maxDegrC And DegrC < (maxDegrC + DegrExtreme) Then
            stbDegrC.ForeColor = System.Drawing.Color.Red
            stbDegrC.Text = "High Temp. Warning: " & DegrC & " degrees Celcius"
        ElseIf minDegrC <= DegrC And DegrC <= maxDegrC Then
            stbDegrC.ForeColor = System.Drawing.Color.Black
            stbDegrC.Text = DegrC & " degrees Celcius"
        Else
            DIGITAL.WriteLine("SOUR:VOLT 0.0, (@204)")
            btnSHOCK.Enabled = False
            btnSTOP.Enabled = False
            btnSTAT.Enabled = False
            btnSTART.Enabled = False
            btnCHANGE.Enabled = False
            tmrDEGR.Enabled = False
            stbDegrC.ForeColor = System.Drawing.Color.Red
            stbDegrC.Text = "Warning: Temperature is outside allowed range."
            MessageBox.Show("Temperature is outside the allowed range. Please fix the problem before restarting
trial.", _
                "TEMPERATURE ALARM", MessageBoxButtons.OK, MessageBoxIcon.Stop)
        End If

        'DegrC += 0.1

    End Sub

#End Region

End Class
```

Appendix C: Visual Basic .NET Code for Classical Conditioning

This program was written by Dawn M. Wendell to allow a person observing the planaria through the camera images on their computer to record the planaria's response during classical conditioning experiments. The responses are recorded in an Excel spreadsheet.

```
Imports Microsoft.Office.Core

Public Class Form1
  Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

  Public Sub New()
    MyBase.New()

    'This call is required by the Windows Form Designer.
    InitializeComponent()

    'Add any initialization after the InitializeComponent() call

  End Sub

  'Form overrides dispose to clean up the component list.
  Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
    If disposing Then
      If Not (components Is Nothing) Then
        components.Dispose()
      End If
    End If
    MyBase.Dispose(disposing)
  End Sub

  'Required by the Windows Form Designer
  Private components As System.ComponentModel.IContainer

  'NOTE: The following procedure is required by the Windows Form Designer
  'It can be modified using the Windows Form Designer.
  'Do not modify it using the code editor.
  Friend WithEvents Label1 As System.Windows.Forms.Label
  Friend WithEvents btnSTART As System.Windows.Forms.Button
  Friend WithEvents txtWORMNUM As System.Windows.Forms.TextBox
  Friend WithEvents Label2 As System.Windows.Forms.Label
  Friend WithEvents Label3 As System.Windows.Forms.Label
  Friend WithEvents LIGHT As System.Windows.Forms.Label
  Friend WithEvents LIGHTSHOCK As System.Windows.Forms.Label
  Friend WithEvents Label6 As System.Windows.Forms.Label
  Friend WithEvents Label7 As System.Windows.Forms.Label
  Friend WithEvents STATUS As System.Windows.Forms.StatusBar
  Friend WithEvents Label5 As System.Windows.Forms.Label
  Friend WithEvents Label8 As System.Windows.Forms.Label
```

```
Friend WithEvents SHOCKvolt As System.Windows.Forms.Label
Friend WithEvents LIGHTvolt As System.Windows.Forms.Label
Friend WithEvents Label11 As System.Windows.Forms.Label
Friend WithEvents Label12 As System.Windows.Forms.Label
Friend WithEvents Label4 As System.Windows.Forms.Label
Friend WithEvents btn1 As System.Windows.Forms.Button
Friend WithEvents btn2 As System.Windows.Forms.Button
Friend WithEvents btn3 As System.Windows.Forms.Button
Friend WithEvents btn4 As System.Windows.Forms.Button
Friend WithEvents tmrREST As System.Windows.Forms.Timer
Friend WithEvents tmrLIGHT As System.Windows.Forms.Timer
Friend WithEvents tmrLIGHTSHOCK As System.Windows.Forms.Timer
Friend WithEvents label9 As System.Windows.Forms.Label
Friend WithEvents Label10 As System.Windows.Forms.Label
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.components = New System.ComponentModel.Container
    Me.btnSTART = New System.Windows.Forms.Button
    Me.txtWORMNUM = New System.Windows.Forms.TextBox
    Me.Label1 = New System.Windows.Forms.Label
    Me.Label2 = New System.Windows.Forms.Label
    Me.Label3 = New System.Windows.Forms.Label
    Me.LIGHT = New System.Windows.Forms.Label
    Me.LIGHTSHOCK = New System.Windows.Forms.Label
    Me.Label6 = New System.Windows.Forms.Label
    Me.Label7 = New System.Windows.Forms.Label
    Me.STATUS = New System.Windows.Forms.StatusBar
    Me.Label5 = New System.Windows.Forms.Label
    Me.Label8 = New System.Windows.Forms.Label
    Me.SHOCKvolt = New System.Windows.Forms.Label
    Me.LIGHTvolt = New System.Windows.Forms.Label
    Me.Label11 = New System.Windows.Forms.Label
    Me.Label12 = New System.Windows.Forms.Label
    Me.btn1 = New System.Windows.Forms.Button
    Me.btn2 = New System.Windows.Forms.Button
    Me.btn3 = New System.Windows.Forms.Button
    Me.btn4 = New System.Windows.Forms.Button
    Me.Label4 = New System.Windows.Forms.Label
    Me.tmrREST = New System.Windows.Forms.Timer(Me.components)
    Me.tmrLIGHT = New System.Windows.Forms.Timer(Me.components)
    Me.tmrLIGHTSHOCK = New System.Windows.Forms.Timer(Me.components)
    Me.label9 = New System.Windows.Forms.Label
    Me.Label10 = New System.Windows.Forms.Label
    Me.SuspendLayout()
    '
    'btnSTART
    '
    Me.btnSTART.Location = New System.Drawing.Point(112, 8)
    Me.btnSTART.Name = "btnSTART"
    Me.btnSTART.TabIndex = 0
    Me.btnSTART.Text = "Start"
    '
    'txtWORMNUM
    '
    Me.txtWORMNUM.Location = New System.Drawing.Point(72, 8)
    Me.txtWORMNUM.Name = "txtWORMNUM"
    Me.txtWORMNUM.Size = New System.Drawing.Size(24, 20)
```

```
Me.txtWORMNUM.TabIndex = 1
Me.txtWORMNUM.Text = ""
'
'Label1
'
Me.Label1.Location = New System.Drawing.Point(8, 16)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(64, 16)
Me.Label1.TabIndex = 2
Me.Label1.Text = "Planaria #"
'
'Label2
'
Me.Label2.Location = New System.Drawing.Point(32, 48)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(80, 16)
Me.Label2.TabIndex = 3
Me.Label2.Text = "Light Only time"
'
'Label3
'
Me.Label3.Location = New System.Drawing.Point(32, 72)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(80, 16)
Me.Label3.TabIndex = 4
Me.Label3.Text = "Light + Shock"
'
'LIGHT
'
Me.LIGHT.BackColor = System.Drawing.SystemColors.ActiveCaptionText
Me.LIGHT.Location = New System.Drawing.Point(112, 48)
Me.LIGHT.Name = "LIGHT"
Me.LIGHT.Size = New System.Drawing.Size(16, 16)
Me.LIGHT.TabIndex = 5
Me.LIGHT.Text = "3"
'
'LIGHTSHOCK
'
Me.LIGHTSHOCK.BackColor = System.Drawing.SystemColors.ActiveCaptionText
Me.LIGHTSHOCK.Location = New System.Drawing.Point(112, 72)
Me.LIGHTSHOCK.Name = "LIGHTSHOCK"
Me.LIGHTSHOCK.Size = New System.Drawing.Size(16, 16)
Me.LIGHTSHOCK.TabIndex = 6
Me.LIGHTSHOCK.Text = "1"
'
'Label6
'
Me.Label6.Location = New System.Drawing.Point(128, 48)
Me.Label6.Name = "Label6"
Me.Label6.Size = New System.Drawing.Size(72, 16)
Me.Label6.TabIndex = 7
Me.Label6.Text = "seconds"
'
'Label7
'
Me.Label7.Location = New System.Drawing.Point(128, 72)
```

```
Me.Label7.Name = "Label7"
Me.Label7.Size = New System.Drawing.Size(72, 16)
Me.Label7.TabIndex = 8
Me.Label7.Text = "second"
'
'STATUS
'
Me.STATUS.Location = New System.Drawing.Point(0, 250)
Me.STATUS.Name = "STATUS"
Me.STATUS.Size = New System.Drawing.Size(292, 16)
Me.STATUS.TabIndex = 9
'
'Label5
'
Me.Label5.Location = New System.Drawing.Point(128, 128)
Me.Label5.Name = "Label5"
Me.Label5.Size = New System.Drawing.Size(72, 16)
Me.Label5.TabIndex = 15
Me.Label5.Text = "volts"
'
'Label8
'
Me.Label8.Location = New System.Drawing.Point(128, 104)
Me.Label8.Name = "Label8"
Me.Label8.Size = New System.Drawing.Size(72, 16)
Me.Label8.TabIndex = 14
Me.Label8.Text = "volts"
'
'SHOCKvolt
'
Me.SHOCKvolt.BackColor = System.Drawing.SystemColors.ActiveCaptionText
Me.SHOCKvolt.Location = New System.Drawing.Point(112, 128)
Me.SHOCKvolt.Name = "SHOCKvolt"
Me.SHOCKvolt.Size = New System.Drawing.Size(16, 16)
Me.SHOCKvolt.TabIndex = 13
Me.SHOCKvolt.Text = "6"
'
'LIGHTvolt
'
Me.LIGHTvolt.BackColor = System.Drawing.SystemColors.ActiveCaptionText
Me.LIGHTvolt.Location = New System.Drawing.Point(112, 104)
Me.LIGHTvolt.Name = "LIGHTvolt"
Me.LIGHTvolt.Size = New System.Drawing.Size(16, 16)
Me.LIGHTvolt.TabIndex = 12
Me.LIGHTvolt.Text = "9"
'
'Label11
'
Me.Label11.Location = New System.Drawing.Point(32, 128)
Me.Label11.Name = "Label11"
Me.Label11.Size = New System.Drawing.Size(80, 16)
Me.Label11.TabIndex = 11
Me.Label11.Text = "Shock Voltage"
'
'Label12
'
```

```
Me.Label12.Location = New System.Drawing.Point(32, 104)
Me.Label12.Name = "Label12"
Me.Label12.Size = New System.Drawing.Size(80, 16)
Me.Label12.TabIndex = 10
Me.Label12.Text = "Light Voltage"
'
'btn1
'
Me.btn1.Location = New System.Drawing.Point(8, 192)
Me.btn1.Name = "btn1"
Me.btn1.Size = New System.Drawing.Size(64, 48)
Me.btn1.TabIndex = 16
Me.btn1.Text = "smooth sliding"
'
'btn2
'
Me.btn2.Location = New System.Drawing.Point(80, 192)
Me.btn2.Name = "btn2"
Me.btn2.Size = New System.Drawing.Size(64, 48)
Me.btn2.TabIndex = 17
Me.btn2.Text = "no motion"
'
'btn3
'
Me.btn3.Location = New System.Drawing.Point(152, 192)
Me.btn3.Name = "btn3"
Me.btn3.Size = New System.Drawing.Size(64, 48)
Me.btn3.TabIndex = 18
Me.btn3.Text = "curls into ball"
'
'btn4
'
Me.btn4.Location = New System.Drawing.Point(224, 192)
Me.btn4.Name = "btn4"
Me.btn4.Size = New System.Drawing.Size(64, 48)
Me.btn4.TabIndex = 19
Me.btn4.Text = "head thrashing"
'
'Label4
'
Me.Label4.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label4.Location = New System.Drawing.Point(8, 168)
Me.Label4.Name = "Label4"
Me.Label4.Size = New System.Drawing.Size(200, 16)
Me.Label4.TabIndex = 20
Me.Label4.Text = "Response During Light Only"
'
'tmrREST
'
Me.tmrREST.Interval = 1000
'
'tmrLIGHT
'
'tmrLIGHTSHOCK
'
```

```
    'label9
    '
    Me.label9.BackColor = System.Drawing.SystemColors.ActiveCaptionText
    Me.label9.Location = New System.Drawing.Point(208, 16)
    Me.label9.Name = "label9"
    Me.label9.Size = New System.Drawing.Size(24, 16)
    Me.label9.TabIndex = 21
    Me.label9.Text = "25"
    '
    'Label10
    '
    Me.Label10.Location = New System.Drawing.Point(232, 16)
    Me.Label10.Name = "Label10"
    Me.Label10.Size = New System.Drawing.Size(48, 16)
    Me.Label10.TabIndex = 22
    Me.Label10.Text = "Trials"
    '
    'Form1
    '
    Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
    Me.ClientSize = New System.Drawing.Size(292, 266)
    Me.Controls.Add(Me.Label10)
    Me.Controls.Add(Me.label9)
    Me.Controls.Add(Me.Label4)
    Me.Controls.Add(Me.btn4)
    Me.Controls.Add(Me.btn3)
    Me.Controls.Add(Me.btn2)
    Me.Controls.Add(Me.btn1)
    Me.Controls.Add(Me.Label5)
    Me.Controls.Add(Me.Label8)
    Me.Controls.Add(Me.SHOCKvolt)
    Me.Controls.Add(Me.LIGHTvolt)
    Me.Controls.Add(Me.Label11)
    Me.Controls.Add(Me.Label12)
    Me.Controls.Add(Me.STATUS)
    Me.Controls.Add(Me.Label7)
    Me.Controls.Add(Me.Label6)
    Me.Controls.Add(Me.LIGHTSHOCK)
    Me.Controls.Add(Me.LIGHT)
    Me.Controls.Add(Me.Label3)
    Me.Controls.Add(Me.Label2)
    Me.Controls.Add(Me.Label1)
    Me.Controls.Add(Me.txtWORMNUM)
    Me.Controls.Add(Me.btnSTART)
    Me.Name = "Planaria Conditioning"
    Me.Text = "Planaria Conditioning"
    Me.ResumeLayout(False)

  End Sub

#End Region

  ' This program conditions planaria using a light-shock method of stimulation for
  ' Classical conditioning.  Intended for use with a planaria training environment
  ' designed by Dawn Wendell from 6/03 - 4/04.
```

```
#Region "Variables and Declarations"

   Public DIGITAL As Agilent.TMFramework.InstrumentIO.DirectIO
   Private COUNTER As Integer = 1
   Private tmrRESTCount As Integer = 30

   Dim oXL As Excel.Application
   Dim oWB As Excel.Workbook
   Dim oSheet As Excel.Worksheet
   Dim oRng As Excel.Range
   Dim oChart As Excel.Chart
   Dim oSeries As Excel.Series

#End Region

   Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load

      'setup, definitions, clearing old info

      tmrREST.Enabled = False
      tmrREST.Interval = 30000
      tmrLIGHT.Enabled = False
      tmrLIGHT.Interval = LIGHT.Text * 1000
      tmrLIGHTSHOCK.Enabled = False
      tmrLIGHTSHOCK.Interval = LIGHTSHOCK.Text * 1000
      COUNTER = 0

      btnSTART.Enabled = True
      btn1.Enabled = False
      btn2.Enabled = False
      btn3.Enabled = False
      btn4.Enabled = False

      STATUS.Text = "Click START to begin testing"

   End Sub

   Private Sub btnSTART_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
btnSTART.Click

      'starts the trial

      tmrREST.Interval = 1000
      tmrREST.Enabled = True
      btnSTART.Enabled = False

      'Excel initialization

      'Start Excel and get Application object
      oXL = CreateObject("Excel.Application")
      oXL.WindowState = Excel.XlWindowState.xlMinimized
      oXL.Visible = True

      'Get a new workbook.
      oWB = oXL.Workbooks.Add
      oSheet = oWB.ActiveSheet
```

```
'Sets Column Titles
oSheet.Cells(1, 1).value = "Trial Number"
oSheet.Cells(1, 2).value = "Response Code"

'Format titles
With oSheet.Range("A1", "B1")
  .Font.Bold = True
  .VerticalAlignment = Excel.XlVAlign.xlVAlignCenter
End With

'Records Settings
oSheet.Cells(2, 6).value = "Planaria Number"
oSheet.Cells(2, 7).value = txtWORMNUM.Text
oSheet.Cells(3, 6).value = "Light Voltage"
oSheet.Cells(3, 7).value = LIGHTvolt.Text
oSheet.Cells(3, 8).value = "volts"
oSheet.Cells(4, 6).value = "Shock Voltage"
oSheet.Cells(4, 7).value = SHOCKvolt.Text
oSheet.Cells(4, 8).value = "volts"
oSheet.Cells(5, 6).value = "Light Time"
oSheet.Cells(5, 7).value = LIGHT.Text
oSheet.Cells(5, 8).value = "sec"
oSheet.Cells(6, 6).value = "Light & Shock Time"
oSheet.Cells(6, 7).value = LIGHTSHOCK.Text
oSheet.Cells(6, 8).value = "sec"

oRng = oSheet.Range("F2", "G6")
With oRng.Borders(Excel.XlBordersIndex.xlEdgeBottom)
  .LineStyle = Excel.XlLineStyle.xlContinuous
  .Weight = Excel.XlBorderWeight.xlThin
End With
With oRng.Borders(Excel.XlBordersIndex.xlEdgeTop)
  .LineStyle = Excel.XlLineStyle.xlContinuous
  .Weight = Excel.XlBorderWeight.xlThin
End With
With oRng.Borders(Excel.XlBordersIndex.xlEdgeLeft)
  .LineStyle = Excel.XlLineStyle.xlContinuous
  .Weight = Excel.XlBorderWeight.xlThin
End With
With oRng.Borders(Excel.XlBordersIndex.xlEdgeRight)
  .LineStyle = Excel.XlLineStyle.xlContinuous
  .Weight = Excel.XlBorderWeight.xlThin
End With
With oSheet.Range("G2", "G6")
  .HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
End With
With oSheet.Range("A2", "B50")
  .HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
End With

oRng = oSheet.Range("A1", "F50")
oRng.EntireColumn.AutoFit()
oRng.WrapText = True

End Sub
```

```
#Region "Timer Ticks"

    Private Sub tmrREST_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles tmrREST.Tick

        'rest timer turns off when 30 sec has elapsed
        If tmrRESTCount = 20 Then
            btn1.Enabled = False
            btn2.Enabled = False
            btn3.Enabled = False
            btn4.Enabled = False
        End If

        If tmrRESTCount = 0 Then
            tmrREST.Enabled = False
            COUNTER = COUNTER + 1
            tmrRESTCount = 30
            tmrLIGHT.Enabled = True

            'TURN ON LIGHT
            ' This code creates and initializes an instance of the DirectIO class
            ' and uses hard-coded values for driver settings.

            DIGITAL = New Agilent.TMFramework.InstrumentIO.DirectIO("GPIB0::9::INSTR", False, 2000000)
            DIGITAL.Timeout = 2000000  'timeout in milliseconds

            DIGITAL.WriteLine("SOUR:VOLT " & LIGHTvolt.Text & ", (@205)")

            STATUS.Text = "Trial #" & COUNTER & ": Light On"

        Else : STATUS.Text = "Rest Interval - " & tmrRESTCount & " seconds remaining"
            tmrRESTCount = tmrRESTCount - 1

        End If

    End Sub

    Private Sub tmrLIGHT_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles tmrLIGHT.Tick

        'turns off light timer and starts light & shock and associated timer
        tmrLIGHT.Enabled = False

        'TURN ON LIGHT and SHOCK
        ' This code creates and initializes an instance of the DirectIO class
        ' and uses hard-coded values for driver settings.

        DIGITAL = New Agilent.TMFramework.InstrumentIO.DirectIO("GPIB0::9::INSTR", False, 2000000)
        DIGITAL.Timeout = 2000000  'timeout in milliseconds

        DIGITAL.WriteLine("SOUR:VOLT " & LIGHTvolt.Text & ", (@205)")
        DIGITAL.WriteLine("SOUR:VOLT " & SHOCKvolt.Text & ", (@204)")

        'Allow user to input the planaria's response
        btn1.Enabled = True
        btn2.Enabled = True
        btn3.Enabled = True
```

```
        btn4.Enabled = True

        tmrLIGHTSHOCK.Enabled = True
        STATUS.Text = "Trial #" & COUNTER & ": Light & Shock On"

    End Sub

    Private Sub tmrLIGHTSHOCK_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles
tmrLIGHTSHOCK.Tick

        'turns off light and shock, turns off associated timer, turns on rest interval timer
        tmrLIGHTSHOCK.Enabled = False

        'TURN OFF LIGHT and SHOCK
        DIGITAL.WriteLine("SOUR:VOLT 0.0, (@205)")
        DIGITAL.WriteLine("SOUR:VOLT 0.0, (@204)")

        If COUNTER >= 25 Then
            STATUS.Text = "Trial Finished. Press START to begin a new trial"

        Else
            tmrREST.Enabled = True

        End If

    End Sub

#End Region

#Region "Response Buttons"

    Private Sub btn1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btn1.Click

        'record in Excel
        oSheet.Cells(COUNTER + 1, 1).value = COUNTER
        oSheet.Cells(COUNTER + 1, 2).value = btn1.Text

        'disable all buttons so a second answer can't be recorded
        btn1.Enabled = False
        btn2.Enabled = False
        btn3.Enabled = False
        btn4.Enabled = False

    End Sub

    Private Sub btn2_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btn2.Click

        'record in Excel
        oSheet.Cells(COUNTER + 1, 1).value = COUNTER
        oSheet.Cells(COUNTER + 1, 2).value = btn2.Text

        'disable all buttons so a second answer can't be recorded
        btn1.Enabled = False
        btn2.Enabled = False
        btn3.Enabled = False
        btn4.Enabled = False
```

```
    End Sub

    Private Sub btn3_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btn3.Click

        'record in Excel
        oSheet.Cells(COUNTER + 1, 1).value = COUNTER
        oSheet.Cells(COUNTER + 1, 2).value = btn3.Text

        'disable all buttons so a second answer can't be recorded
        btn1.Enabled = False
        btn2.Enabled = False
        btn3.Enabled = False
        btn4.Enabled = False

    End Sub

    Private Sub btn4_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btn4.Click

        'record in Excel
        oSheet.Cells(COUNTER + 1, 1).value = COUNTER
        oSheet.Cells(COUNTER + 1, 2).value = btn4.Text

        'disable all buttons so a second answer can't be recorded
        btn1.Enabled = False
        btn2.Enabled = False
        btn3.Enabled = False
        btn4.Enabled = False

    End Sub

#End Region

End Class
```