2021

The implementation of a novel graphical python editor (STREMECODER) in a rodent discrimination apparatus

Supraja Kalva

The implementation of a novel graphical python editor (STREMECODER) in a rodent discrimination apparatus

Abstract

As the interest in neuroscience and the desire to perform behavioral tasks in a higher level of specificity and accuracy increases, the need to have tools and techniques to conduct experimentations in a low-cost automated manner is essential. Although such methods have been proposed previously by other researchers, they have presented their data and tools in a manner that would have been difficult to comprehend for non-programmers. In labs that do not have the accessibility to individuals who can understand the published procedures, it is very difficult for them to get started and manipulate the published procedures to their interests. Here in this paper, I discuss designing and implementing a computer program for a rodent odor discrimination apparatus using a novel graphical python programming editor named STREMECODER. The objective of this project is to create a program interface that helps the laboratory staff conduct their experiments in an automated, dynamic, and userfriendly manner.

Degree Type

Open Access Senior Honors Thesis

Department

Computer Science

First Advisor

William Sverdlik

Second Advisor

S. Maniccam

Third Advisor

Tom Mast

Subject Categories

Computer Sciences | Neuroscience and Neurobiology

THE IMPLEMENTATION OF A NOVEL GRAPHICAL PYTHON EDITOR (STREMECODER) IN A RODENT ODOR DISCRIMINATION APPARATUS

By

Supraja Kalva

A Senior Thesis Submitted to the

Eastern Michigan University

Honors College

in Partial Fulfillment of the Requirements for Graduation

with Honors in Computer Science and Neuroscience

Approved at Ypsilanti, Michigan, on this date 04/25/2021

Computer Science Supervising Instructor: ___ _ Date: 4/26/2021____

Computer Science Departmental Honors Advisor: S.Maniccam Date: 4/26/2021

Computer Science Department Head: Date: 4/26/2021____

Neuroscience Supervising Instructor: ___ Date: 4/26/2021___

Neuroscience Program Honors Advisor: ___ Date: 26 April 2021

Neuroscience Program Head: _____ Date: 26 April 2021

Dean, Honors College: _ Date: 26 April 2021

TABLE OF CONTENTS

ABSTRACT

As the interest in neuroscience and the desire to perform behavioral tasks in a higher level of specificity and accuracy increases, the need to have tools and techniques to conduct experimentations in a low-cost automated manner is essential. Although such methods have been proposed previously by other researchers, they have presented their data and tools in a manner that would have been difficult to comprehend for non-programmers. In labs that do not have the accessibility to individuals who can understand the published procedures, it is very difficult for them to get started and manipulate the published procedures to their interests. Here in this paper, I discuss designing and implementing a computer program for a rodent odor discrimination apparatus using a novel graphical python programming editor named STREMECODER. The objective of this project is to create a program interface that helps the laboratory staff conduct their experiments in an automated, dynamic, and user-friendly manner.

INTRODUCTION

BACKGROUND

As mentioned by a recently published Arduino project, behavioral testing of rodents is a key category of experiments in clinical research, providing a tool to test, assess, and develop treatments for patients in need with various diseases and disorders [1]. Rodents are often used as a subject in most behavioral tests for neuroscience and other disciplines as the similarities between them and the human genome are quite remarkable, with most genes of one species occurring in the other [11]. To perform such behavioral experiments with rodents, there are commercially available apparatuses with specific software that laboratories could purchase. However, many of these behavioral tests and equipment can be exorbitantly high-priced, making it difficult for smaller laboratories to purchase them which limits the accessibility ratio of these products in smaller laboratories [1].

To combat this, smaller laboratories have been getting more involved in implementing systems that use a low-cost and open-source I/O board (Arduino family). Arduino is an open-source physical computing platform founded on a simple I/O board and a development environment that implements the Processing/Wiring language [2]. The Arduino could be used to develop independent interactive objects or could be connected to the software on your computer [2]. The open-source integrated development environment (IDE) can be downloaded for free (currently available for Mac OS X, Windows, and Linux) [2].

Due to its wide range of usage, the amount of published research that integrates the Arduino board in their various types of behavioral testing systems is extraordinary [3, 4, 5, 7, 8, 10]. Although these publications exist and provide detailed information regarding their set-up and usage, for smaller laboratories that might not have the appropriate resources to get help on how to get started or manipulate the existing code and procedures for their experiments, this becomes a big problem. To help mitigate the

number of problems a potential laboratory could face while attempting to understand the complexity of various published codes, we are working on implementing a graphical programming editor, named, STREMECODER [6].

STREMECODER

STREMECODER works similarly to an IDE and implements three unique features (documentation, form, and nodes/threading/looping) in one platform [6]. The diagrammatic visual representation within the editor makes it easier for an outsider with no knowledge of the experiment to understand what exactly is happening in each step of the program code. Essentially, a user can create coherent groups within the code so that it is easy to understand and debug the program flow while the user works on developing the program, see Figure 1.

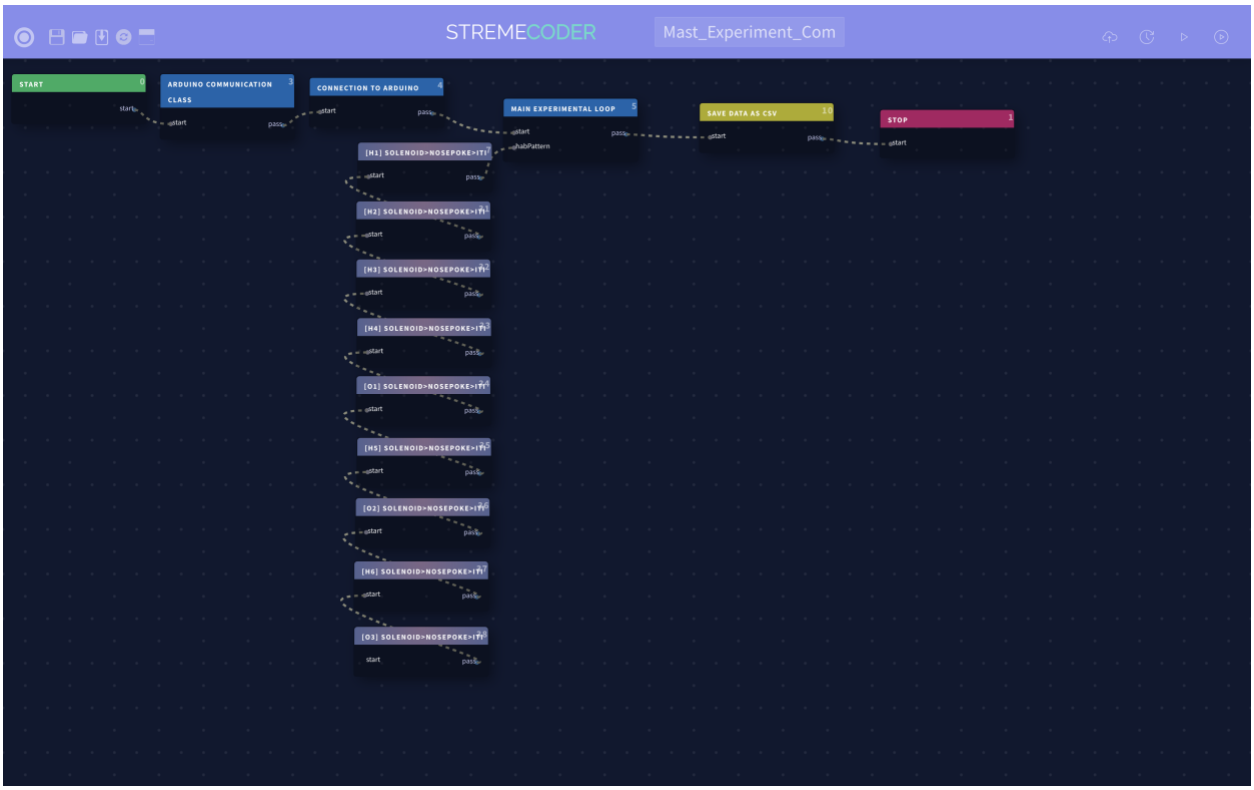


Figure 1: A diagrammatic visual representation of the program

The documentation feature of STREMECODER is unlike any other editors that are of the popular choice of interest by most programmers. The documentation feature allows the user to document a specific section of code, node, using a markdown editor, similar to the one seen on GITHUB, which is a cloud-based Git repository hosting service. This is particularly useful for laboratories because there are often multiple people who might want to replicate the code and manipulate/develop it differently and the best way to do that is by providing good documentation so that others can clearly understand what is happening inside your code. Consequently, the form feature allows the user to enter information that can be later used as variables in the code. This allows easy and quick manipulation of different variables that are often dynamic during experimentation such as trial time or trial name, as seen in Figure 2.

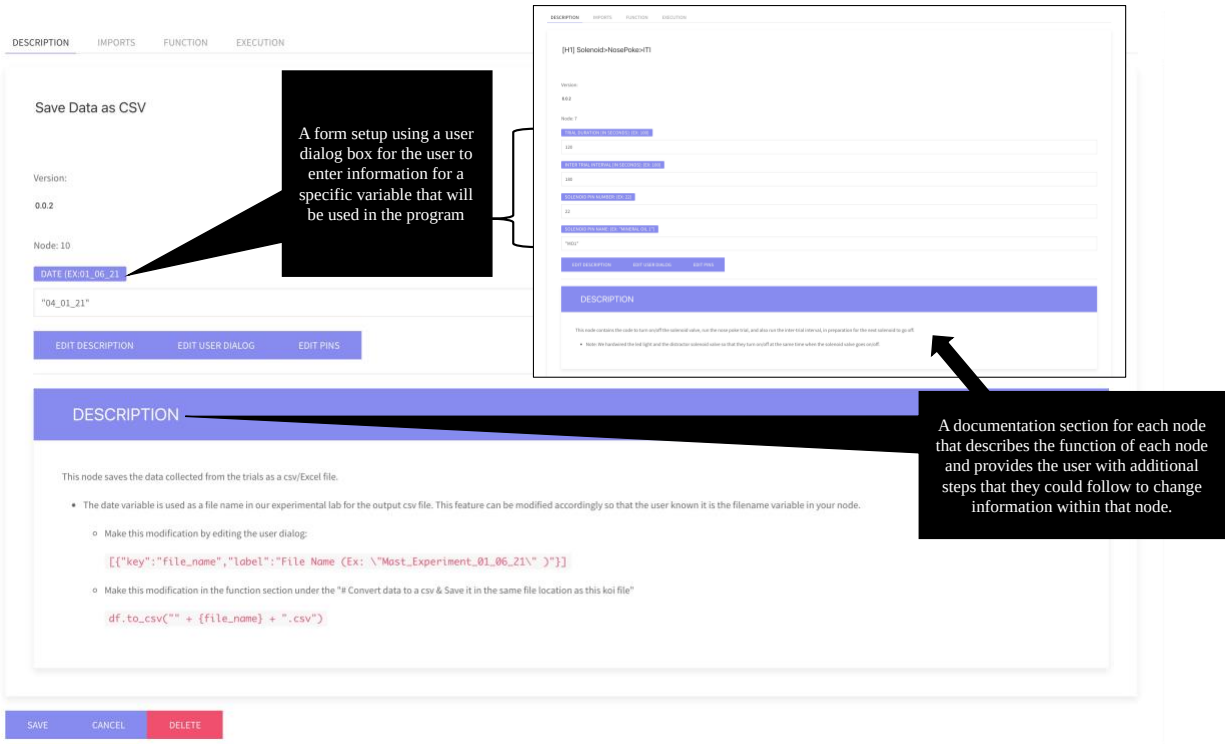


Figure 2: A diagram that displays a deeper look of two nodes referenced in Figure 1 that explains how the documentation and form features of STREMECODER are represented.

The looping/threading feature is the most exciting of the editor. The looping feature allows you to run a certain section of nodes multiple times by just calling one variable. As seen in Figure 3, the day variable, or pin in STREMECODER, allows the user to run the day loop that contains the nodes eat,

sleep, and enjoy, in that order, multiple times based on the number of times the variable is referenced in the person node. A node, as referenced multiple times earlier, is a basic unit of data structure. Nodes contain data that is linked to other nodes to create complex data structures. In STREMECODER, the node is a certain section of code that is linked to a beginning node and ending node, similar to nodes seen in a linked list. Just as linked lists, multiple nodes could be connected as one line of nodes acting as a thread, with each connecting node being a function of the next node, for example, Node4(Node3(Node2(Node1)))). The starting and ending nodes decide where a certain thread begins and ends making it easier to create and test program flow.

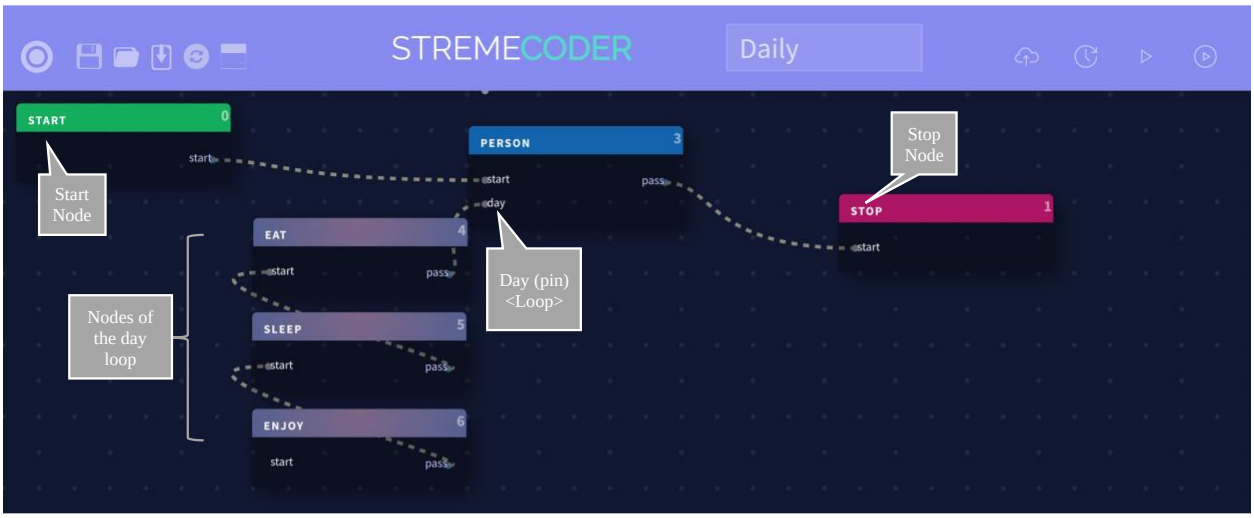


Figure 3: An example that demonstrates the flow of a STREMECODER program

OBJECTIVE

Essentially, in this project, I designed, constructed, and validated a program using STREMECODER for a rodent odor discrimination apparatus that was described in a recently published paper [9]. In this recently published paper, the authors describe a “versatile and automated setup, “Poking-Registered Olfactory Behavior Evaluation System” (PROBES), which can be adapted to perform multiple olfactory assays” [9]. The apparatus that was used in this project focuses on using the PROBES system to determine odor thresholds in naïve animals. The program that was written for this apparatus

will provide an automated, dynamic, and user-friendly application to evaluate these innate odor-triggered behaviors of the animal.

Code

Schematics

The schematics used to develop the program are established in the Appendix, please review them for more information regarding the timeline of this project, the design of the program, and much more.

GitHub Repository

The program code has been published to a public GITHUB repository that can be accessed using this URL: https://github.com/SuprajaKalva/Mast-Lab-PROBES.

Discussion and Future Directions

Due to the COVID-19 situation, many hurdles, such as delays in testing, were conquered for the completion of this project. The project was conducted following the rules and regulations established by the university as well as the State of Michigan to practice social distancing and avoid exposure to the virus. Overall, the project was reasonably successful. The laboratory staff, especially the lab supervisor, were able to modify the program effectively parallel to the various experimentations and trials that were being executed. In particular, they were able to modify the necessary nodes with input and run the program in a relatively swift manner, and with the errors logged in the terminal, they knew exactly when the program has failed to do its particular task. This was extremely helpful while testing the developed program with the apparatus.

In addition to the already implemented features, a few other features that can be added to, or enhanced in the program for future improvements are listed below.

1. A mechanism in which all the sensors could be reset to their initial states, in case the automated program needs to be terminated manually.

2. An enhanced approach to aggregate and analyze the data from the sensors in the apparatus than the outputted CSV file. Essentially, adding additional nodes in the program so that the program can automatically output graphs and charts using the data generated by the program. The graphs and charts could be created using libraries such as Matplotlib and Plotly. As the scalability of the data increases, using Big Data tools to aggregate the data could be beneficial, primarily when sensor data is extensive for more prolonged experimental trials.

All in all, this project was conducted to provide a platform in which smaller laboratories could efficiently develop and modify code for their applications in a reduced amount of time and learning period. As a non-programmer, a laboratory staff would be able to use this program to modify the program with the necessary variables for the day's trial with minimal help or explanation on what to do. In the long run, this will save a lot of time as the laboratory staff can spend less time learning and more time doing research.

Additionally, in the broader sense, research projects such as this are essential now more so than ever because more and more laboratories are creating apparatuses with Arduinos. Using editors like this will allow researchers to share their projects more conveniently. Furthermore, projects like this will allow a wider range of users to conduct behavioral testing of mice due to the multifaceted nature of Arduino. Moreover, carrying out projects like this facilitates the progress of computer science and neuroscience research in a more integrated manner, allowing programmers and developers to use their skills to create,

innovate, and inspire across disciplines. If more questions are being answered regarding one stimulus they can be easily manipulated for other stimuli.

Essentially, this project discusses an Arduino and Python-based platform to control a rodent odor discrimination paradigm. Using an Arduino micro-controller and Python-based custom-written program using STREMECODER, a defined stream of odor can be delivered to the animal depending on the timing and number of nose pokes are reported. This system facilitates the end-user to manage the behavioral testing in real-time and hence contributes a robust custom-made platform for probing innate odor-triggered behaviors of the animal.

REFERENCES

1. *A Behavioural Chamber to Evaluate Rodent Forelimb Grasping*. (n.d.). Arduino Project Hub. Retrieved April 14, 2021, from https://create.arduino.cc/projecthub/alejandrocarn/a-behavioural-chamber-to-evaluate-rodent-forelimb-grasping-bedb1a
2. *Arduino—HomePage2*. (n.d.). Retrieved April 14, 2021, from https://www.arduino.cc/en/Main/HomePage2
3. Chen, X., & Li, H. (2017). ArControl: An Arduino-Based Comprehensive Behavioral Platform with Real-Time Performance. *Frontiers in Behavioral Neuroscience*, *11*. https://doi.org/10.3389/fnbeh.2017.00244
4. D'Ausilio, A. (2012). Arduino: A low-cost multipurpose lab equipment. *Behavior Research Methods*, *44*(2), 305–313. https://doi.org/10.3758/s13428-011-0163-z
5. Devarakonda, K., Nguyen, K. P., & Kravitz, A. V. (2016). ROBucket: A low cost operant chamber based on the Arduino microcontroller. *Behavior Research Methods*, *48*(2), 503–509. https://doi.org/10.3758/s13428-015-0603-2
6. *Graphical Python Programming Editor*. (n.d.). StremeCoder. Retrieved April 14, 2021, from https://www.stremecoder.com
7. Micallef, A. H., Takahashi, N., Larkum, M. E., & Palmer, L. M. (2017). A Reward-Based Behavioral Platform to Measure Neural Activity during Head-Fixed Behavior. *Frontiers in Cellular Neuroscience*, *11*. https://doi.org/10.3389/fncel.2017.00156
8. O'Leary, J. D., O'Leary, O. F., Cryan, J. F., & Nolan, Y. M. (2018). A low-cost touchscreen operant chamber using a Raspberry Pi™. *Behavior Research Methods*, *50*(6), 2523–2530. https://doi.org/10.3758/s13428-018-1030-y
9. Qiu, Q., Scott, A., Scheerer, H., Sapkota, N., Lee, D. K., Ma, L., & Yu, C. R. (2014). Automated Analyses of Innate Olfactory Behaviors in Rodents. *PLOS ONE*, *9*(4), e93468. https://doi.org/10.1371/journal.pone.0093468

10. Rizzi, G., Lodge, M. E., & Tan, K. R. (2016). Design and construction of a low-cost nose poke system for rodents. *MethodsX*, *3*, 326–332. https://doi.org/10.1016/j.mex.2016.04.002
11. Wahlsten, D. (2011). Chapter 2—Mice. In D. Wahlsten (Ed.), *Mouse Behavioral Testing* (pp. 15–37). Academic Press. https://doi.org/10.1016/B978-0-12-375674-9.10002-3

Conflict of Interest Statement

As the author, I declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Author's Note

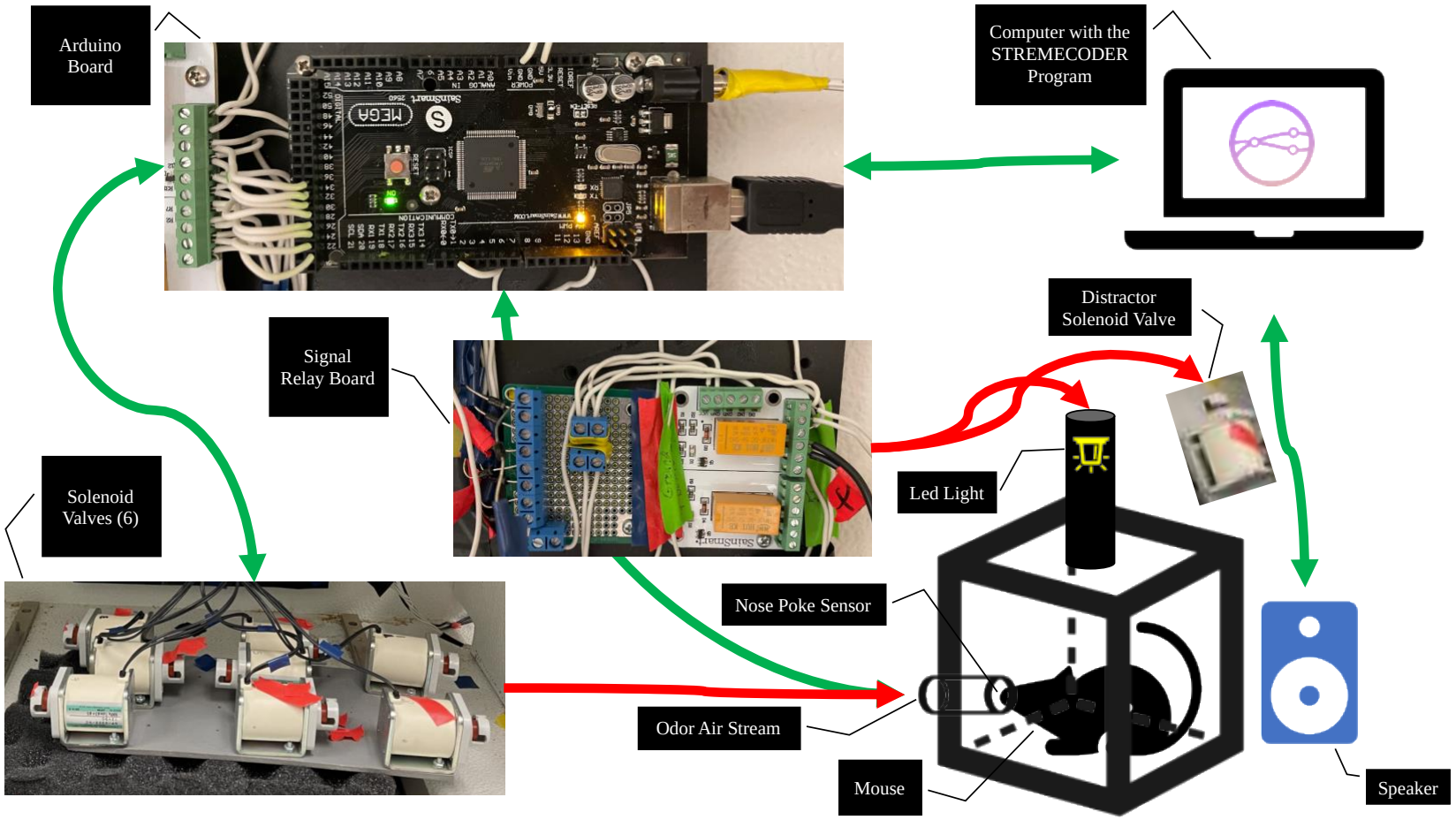
I would like to thank Dr. Dave Peña for all his help with the software while testing and developing the program for this project. I would also like to thank Dr. Mast, Dr. Sverdlik, Dr. Maniccam, and Dr. Evans for their help in revising this manuscript.

APPENDIX

A: PROJECT TIMELINE

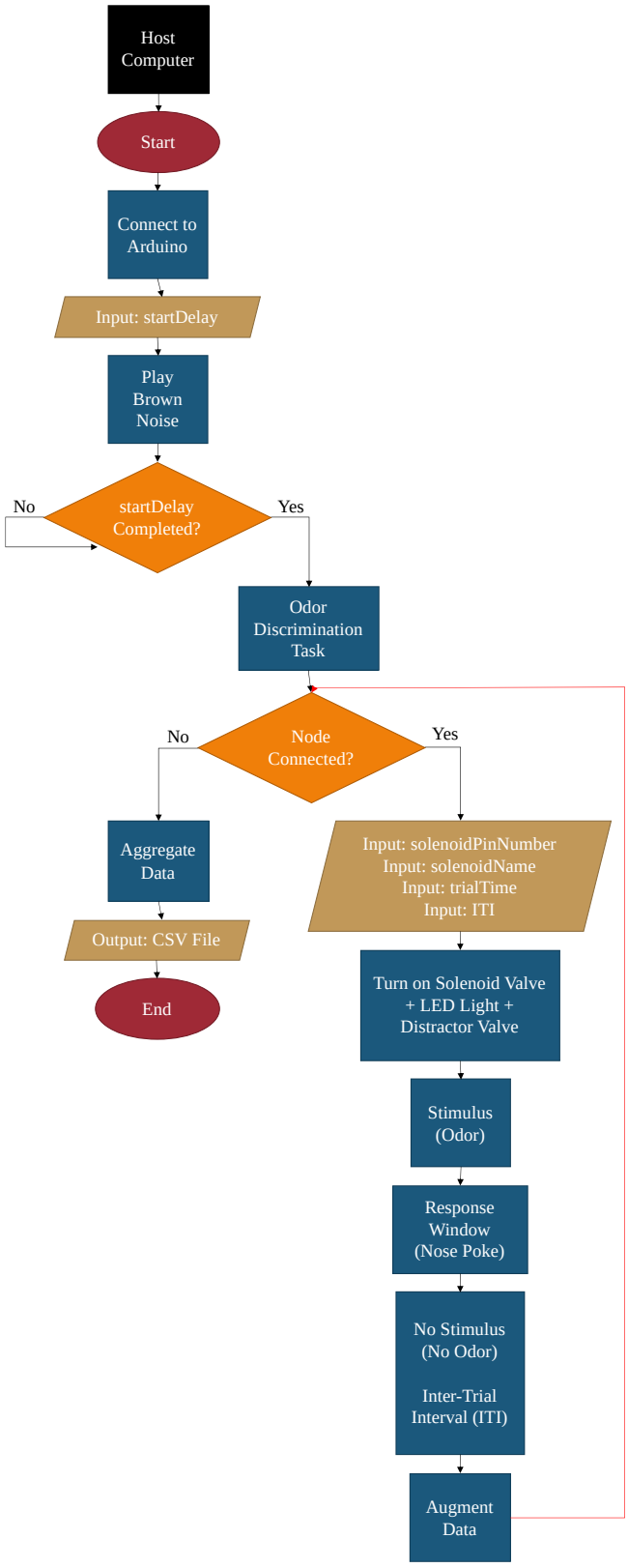
Project Timeline	Aug	Sept	Oct	Nov	Dec	Jan	Feb	Mar	Apr
Research This phase involved gathering and analyzing various algorithms and techniques that were already used and established regarding the use of Arduinos. During this phase, narrowing down on what specific method of implementation and how it will be implemented during the later phases of the study was achieved.	■	■	■						
Planning This phase involved starting to write the pseudo code and organizing the layout of the project clearly so that the execution phase would be taking place smoothly. During this phase, the gathering of how to use STREMECODER, the construction of an interface/algorithm, and testing was achieved.			■	■	■				
Execution This phase involved implementing the algorithm during experimentation. This phase will require the most amount of attention and care as experimentation can only take place once. Therefore, the previous, planning stage must be conducted most efficiently and properly.						■	■	■	
Preparing the manuscript for submission									■

B: Design Schematic of the Setup



This schematic demonstrates how the rodent odor discrimination apparatus was set up. The green arrows represent from where the input and output are being sent and received, while the red arrows represent only where the output is being sent. This apparatus schematic only shows the necessary components needed to develop the program, there are more elements to the apparatus which are not displayed here.

C: Flow chart of the program



Flow chart illustrating the flow of information into and out of the STREMECODE program. The host computer initiates the start node which initiates various processes until the odor discrimination task is started. Once the task is started, it will traverse through all the nodes in an orderly looped fashion. When the traversal is completed, all data from the nodes will be aggregated and will be outputted as a CSV file.