ECE/COSC 402

Final Report

RAT MANN: Rage Against the Multi-level Adaptive Neural Network

Jaidin Jackson, Megan Stanton, Jessica Stevens, Alex Twilla, Sam Zimmermann

Submitted: 4/24/2021

Executive Summary

Neuromorphic computing, a subcategory of artificial intelligence and machine learning, is typically difficult for the everyday person to comprehend, for it is a fairly new field of technology that requires a background in calculus, physics, biology, circuit theory, and programming. The ultimate goal of RATMANN is to provide the everyday person a means to understand the complexities and importance of neuromorphic computing while simultaneously providing a fun and interactive experience. It also aims to spark an interest in those who use it, especially those part of the younger crowd, in the hopes that they may pursue an education in AI/computer engineering and contribute to the field later on in life.

RATMANN's physical design is to reflect that of a two-player arcade game. The game itself will be similar to *Guitar Hero* or *Dance Dance Revolution*. The following are the top five customer requirements: (1) must support a human player and an AI player simultaneously, (2) must demonstrate the network's effectiveness through a scoring system, (3) must have one screen for displaying the game state and a second screen for displaying the score and network topology, (4) must be powered in way that allows for easy portability, and (5) must be visually appealing.

The physical components of RATMANN include a Raspberry Pi, buttons, two LCD screens, speakers, and LEDs. The functions of these components are to serve as the computational core of the entire system, allow for user input to the game, visually display both the game and the neural network, output the game's sounds, and exhibit the neural network's input, respectively. As for the software side, the environment will be built in a format similar to OpenAI Gym environment to allow RATMANN to be more versatile to open source developers within the AI community.

Additionally, this report explains the reasoning behind various design decisions, specifically the decisions to utilize the format of an existing environment and the final physical design and layout of the product. It also provides a detailed timeline of the projected project milestones, a detailed breakdown of the project's budget, and a list of deliverables that should be supplied upon completion of the project. These deliverables include a Guitar Hero OpenAI Gym Application, Electrical Circuitry, Firmware Code, OpenAI Gym/Model Interface, and an "Arcade-style" Housing.

In order to demonstrate how RATMANN has successfully fulfilled the customer requirements/design objectives/engineering characteristics mentioned previously, a set of experiments have been designed and conducted. Specifically, these experiments evaluated the OpenAI Gym Interface and the playability of the game itself. Descriptions of the experiments themselves as well as their successful results can be found in the "Test Plan" section, which also references test matrices listed in the Appendix.

Lastly, this report details the final tangible product through the use of visual aids from various angles and screens, thorough explanations of the modules, the inputs and outputs of each module, as well as how they interact with each other. More specifically, the "Embodiment Design" section goes over the software modules, including the rhythm game environment, the wrapper environments, and the visualization program, as well as the hardware modules, including the Raspberry Pi 4, button LEDs, and other hardware that interacts with the environment. Explanations of code functionality, coding language of choice, file formats (JSON files, step map files), neuromorphic training, how to play RATMANN, can be found here.

It is important to note that the following sections have been updated to reflect the progress that has been made throughout the duration of the semester: "Project Management", "Budget", and "Design Concepts, Evaluation, & Selection". Although the bulk of these sections have been minimally revised since the last semester, for the final product has not strayed much from its original design, they still detail any adjustments to the project and reasons behind these adjustments.

Table of Contents

I. Problem Definition and Background…… 4
II. Requirements Specification…….. 4
III. Technical Approach…….. 5
IV. Design Concepts, Evaluation, & Selection……...……..….. 6
V. Embodiment Design……. 7
VI. Test Plan…… 9
VII. Deliverables……. 10
VIII. Project Management……...……... 10
IX. Budget……...……..... 11
X. References……. 12
XI. Appendix……... 14

List of Tables:

Table Number	Description	Page Number
1	Engineering Characteristics	5
2	Physical Design Decision	7
3	Software Design Decision	7
4	File Format Design Decision	7
5	DANNA2 Implementation Decision	7
6	Project Milestones	10
7	Updated Project Milestones	11
8	Original Budget	11
9	Updated Budget	11

List of Figures:

Figure Number	Description	Page Number
1	A block diagram of the overall system	5
2	A block diagram of the software system	6
3	Two options for the physical design	7
4	RAT MANN Start-Up Game Screen	9
5	RAT MANN Gameplay Screen	9
6	Arcade-Style Housing and Internal Supports	9
7	Electrical Component Placement	9

I. Problem Definition and Background

Artificial intelligence and machine learning (AI and ML) is a relatively new field that can significantly impact how we solve problems in the future. A subset of this field is neuromorphic, or "brain inspired" computing that more closely mimics functionality of the human brain. Due to its comparatively low power consumption due to sparse network topologies, as well as its ability to solve problems not able to be solved in finite time by traditional computer architectures, neuromorphic computing has potential to revolutionize computing and essentially become the new Von Neumann architecture.

This emerging field of neuromorphic computing is incredibly exciting; however, the work done in this field is not digestible to the general public. Neuromorphic computing and AI and ML typically require understanding of high-level calculus, physics, and some level of human biology to truly understand. Additionally, a lot of the direct work being done in this field requires knowledge of circuit theory and programming. Because of this, there are no "entry level" neuromorphic computing resources available like exists for many other disciplines within engineering. Some of the reason for this is because it is such a new field of study with a developmental focus rather than educational. In a broader scope, there are very few learning systems for artificial intelligence and machine learning, although more are popping up online as the field becomes more saturated.

Our technology aims to bridge the gap between experts in neuromorphic computing and the layman. To do this, we plan to create a game demo in the format of a multiplayer game. A participant will play against a neural network on a game in a similar format to *Guitar Hero*. The player will play three rounds; each round, the neural network will progress in its efficiency, to demonstrate learning. The scores from each round between the player and that round's neuromorphic challenger will be compared, and the neuromorphic player's scores across the three rounds will be compared to demonstrate an increase in fitness. The project is being designed for our sponsor, the TENNLab group at the University of Tennessee, Knoxville, to be utilized as a live demonstration in lab tours as well as at conferences and at on-campus events for high school students.

Our target customer would be educators that have access to high school age students, either through college lab tours or, ideally, secondary school classrooms. This work would be an important tool used to grab the interest of younger people, who could potentially become contributors to the computer engineering field later on. Additionally, this project can be used to further motivate the field to potential investors or grantors who may be interested in the work of the research lab. Lastly, this work can be used to interest current college students into working in the TENNLab research group. Current students may be exposed through lab tours, through exposure in lower-division classes, or through their colleagues, that may entice them to pursue undergraduate research in general, or specifically AI and ML.

II. Requirements Specifications

The overall concept of our design is a multiplayer-style arcade game where one player is a person, and the other is an AI. The game will be similar to a combination of *Guitar Hero* and *Dance Dance Revolution*, a timing-based game where you press colored buttons in correspondence to those shown on a screen for both 'players'. There will be multiple rounds, each round using a neuromorphic model of increasing fitness, or accuracy. The score for both the human player and the neuromorphic player will be displayed at the end of each round, and a final tally of all scores will be shown at the end of all three showing who is the better player.

More formally, the requirements specification for this project are as follows:

1. The game must be playable by a human player and an AI player;
2. The game must have a scoring system that demonstrates the effectiveness of the network;
3. The system must be able to be powered by a single wall plug for portability;
4. The system must work together cohesively, so the hardware must successfully interact and affect the game state;
5. The game state must be displayed on the left hand screen using either the pygame or pyglet rendering library, and the score and an image of the current network topology displayed on the right hand screen;
6. Each round of the game must be able to be run manually, with specified arguments such as the song and the DANNA2 network to be used;
7. The arcade game must have a finished and attractive appearance, adequate to be utilized as a laboratory demo or taken to conferences.

Second, the stretch goals for our project, assuming all above goals are satisfied, are as follows:

1. The system will step through the three game rounds on its own without user input;
2. The player will be able to select between multiple songs for each round;
3. The right hand screen will display a live demonstration of the operation of the neural network, instead of a still image.

The requirements for this project can be broken down into several characteristics. Each characteristic is a variable or a constraint, meaning either we must work within the confines

of that characteristic, or we have the ability to modify that aspect of the design to best meet our requirements. This is displayed in further detail in Table 1 below.

Table 1: Engineering characteristics

Engineering Characteristic	Options	Significance
OpenAI Gym Interface	Must follow framework specifications	Constraint: must be written in Python; must follow the OpenAI Gym format
Neuromorphic model	TENNLab DANNA2 model	Constraint: must use the available OpenAI Gym compatible AI model
Playable app	Programming language, game-related details	Variable: can change based on implementation choices in correspondence with the hardware design
Number of players	One human and one neuromorphic player	Constraint: the definition of the project constrains the design to two players: one human and one AI
Output	One or two screens, of same or different sizes; audio output	Variable: the project needs at least one screen; however, the purpose and use of a second screen is optional, and the sizes of each screen can be varied as needed. Additionally, the number of audio output speakers utilized is also variable
Input	Number of buttons for each player	Variable: the number of input buttons per player can vary as long as each player has equal inputs, which will correspond to the game implementation
Microcontroller/ control board	Various microcontroller boards (ie Raspberry Pi, Beaglebone Black)	Variable: constrained by the input/output requirements, and must have computing power to run the application, but otherwise the particular board selected is variable
"Arcade -style" Housing design	Materials used (3D printing, wood, metal, glass, etc), orientation, and configuration	Variable: must accommodate electrical design and have a finished, clean appearance, but is otherwise flexible

III. Technical Approach

As the goal of RAT MANN is to demonstrate the efficacy of neuromorphic technology through a video game, it will take the form of a small arcade cabinet. Within this cabinet are the following components: a Raspberry Pi, a button for playing the game, two LCD screens, speakers, and LEDs that will display the neural network's input. The Raspberry Pi will serve as the computational core of this system; it will simulate the neuromorphic hardware, receive player input, output network response, and host the game environment. This game environment will be built in the style of an environment from OpenAI's Gym library. Doing so is favorable to this system as this format is widely used in testing throughout the AI community, so it will make RATMANN more understandable and easier to work with for any potential open source developers. The left LCD screen will host the game for the player, and the right screen will host a visualization of the network, displaying all of the network's neurons and synapses as well as any firing events so as to give the player and any spectators additional insight into the working of the neuromorphic model. Figure 1 shows a high-level block diagram of the overall RATMANN system.

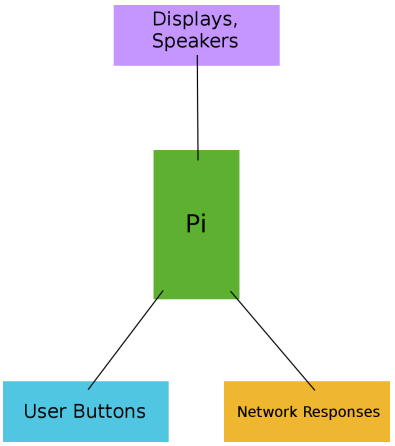

Figure 1: A block diagram of the overall system.

While a hardware implementation of the DANNA2 neuromorphic model is possible, it requires an FPGA, and due to the high cost of FPGAs as well as any potential added complexity, the decision to instead use a simulation of DANNA2 was reached. This may cause the system to not truly be neuromorphic in nature, but the simulations are reliable, and this approach also offers the added benefit of increased centralization as the simulations will be run on the Raspberry Pi. It also removes any need to create and/or implement an interface between an FPGA and the aforementioned Raspberry Pi. Also in the pursuit of simplicity arose the decision to use tactile buttons over capacitive touch input. Tactile buttons -- on top of reducing the costs and implementation capacity -- are also more suitable to the purpose of playing a rhythm game, where tactile feedback on one's input is paramount. Also paramount in rhythm games is acceptable song mapings. To this end, the game environment will utilize a systematically modified version of *Dance Dance Revolution* step map file format. Doing so immediately grants RATMANN access to a vast library of potential songs for use in this project, and it

also increases accessibility to open source developers, as the step map format is well-documented and relatively familiar.

As mentioned previously, the game environment will be built in the format of an OpenAI Gym environment, as this is a standard format for AI testing environments. This environment will represent the rhythm game in a highly abstract form; notes that the network must hit will be represented as bit strings that are given a distance from the "strum area" that continually decreases. This environment will be contained in a higher-level module that will interface with both the neuromorphic network and the user's input. This module will use the data from the environment to create a game render that will be displayed to the user on the left LCD screen. User input and the network's responses to the game will be handled via the Raspberry PI's GPIO pins. A diagram of this software configuration can be found in Figure 2.

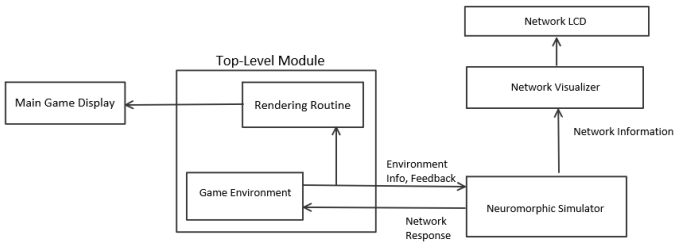

Figure 2: A block diagram of the software system.

IV. Design Concepts, Evaluations, and Selection

Among the major design decisions for this project are the *Guitar Hero*-esque environment design, how to read songs into the environment, using an FPGA implementation of the neuromorphic model or a simulation, the physical structure of the cabinet that contains all of the components, and the sizes of the LCD screens used to display the game and network visualization. The primary options for creating the game environment were to either create a custom environment or build one based on some already established paradigm. The advantage of the former is the ability to tailor the environment specifically to the needs of the application. While this is a notable advantage, is it outweighed by the positives of adhering to the format of some premade environment style -- namely OpenAI Gym. Such advantages consist of already having a tried and true environment framework to follow with ample documentation and examples as well as broad familiarity within the AI community, making open source development considerably more accessible for those who would wish to do so. Given that one of the aims of this project is to invite open source development, the option of building the game environment in the style of an OpenAI Gym environment is the more optimal of the two.

A similar line of reasoning was the impetus behind the next design decision: that of choosing how to read songs into the environment. Again, the principal options for this decision were to create a custom file format or to use an already established format. Creating a file format for a song is quite a tedious process as it requires one to individually map every note that they wish to be included in the game, and depending on the song and/or difficulty, this could require up to hundreds of notes per beat of any given song. Given the time constraints of this project, this option is simply unfeasible. Luckily, however, another option exists in the *Dance Dance Revolution* step map file format. In choosing this option, a plethora of open source step map files becomes accessible, cutting a considerable amount of overhead out of this project. The only modification that this method would require would be to change the step map format to work with five inputs as opposed to four, which can be easily done. Yet another file format exists for this decision: the *Guitar Hero* file format. This would seem intuitive given that *Guitar Hero* is the inspiration of this project, but *Guitar Hero* simply differs too much mechanically with how RATMANN's environment will function. Namely, on top of having to input the correct notes, *Guitar Hero* requires an additional "strum" input that would complicate the environment. There are also a few mechanics that would have to be detected and removed from the song files such as star notes, tapping, and the whammy bar. Simply put, *Dance Dance Revolution*'s format is simpler and matches with the objectives much better that that of *Guitar Hero*, and it eliminates the time cost and tedium of creating an original file format.

Upon reviewing this project's budget, the choice between a physical, FPGA implementation of the neuromorphic model and a simulation of the model became quite a simple one. Given the relatively high cost of an FPGA, our budget simply could not accommodate such an implementation without significant out of pocket expenses. While this financial constraint is a compelling reason to use a simulation, other convincing reasons exist as well. Implementing an FPGA into our system would require more space on the interior of the arcade cabinet, potentially leading to a redesign and remake, which would incur more costs. On top of this downside, an FPGA implementation would also introduce the need to interface with Raspberry Pi, whereas using the simulation would allow for a more centralized system; i.e. everything would be run on the Pi, simplifying the system considerably. Given these reasons, using a simulation of the neuromorphic model seems to be the optimal choice.

When considering the final physical design for our project, we had two concepts, where each concept had a different placement for the two screens. The first concept had the two screens side by side vertically, while the second concept had the two screens side by side horizontally. The difference between the two concepts can be seen in Figure 3. The player buttons will be light up LED so that the player can know if their press is recorded and for visual aesthetic. The neural network will also have the same buttons as the player, but they will not have a function other than to light up as though the neural network is pressing the button. While we

thought of multiple ways to situate the speakers, we inevitably ended up deciding that they should go on the sides of the inside of the box, with slates for the sound to travel through. We decided to put the speakers on the inside of the box in order to avoid any accidental damage. We also had to decide between two different screen sizes. The first option was a seven inch screen with haptic feedback and capacitive touch buttons. The second option was a ten inch screen with light-up arcade buttons. We chose to go with the second option because we determined that our consumers would most likely want a larger screen. The reason we could not do the larger screen with the haptic feedback and capacitive touch was because it did not fit into the budget.

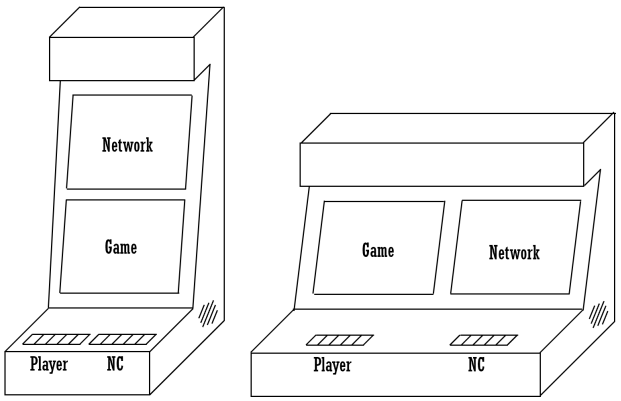

Figure 3: Two primary options for physical design.

Both options fit all defined engineering characteristics. Half of our engineering characteristics pertained to the software aspect of our design which did not have options, based on the aforementioned reasons. The two concepts, horizontal versus vertical, mainly pertain to the "Arcade-style" housing design characteristic. Figure 3 shows the two primary configurations and we chose the horizontal approach because it fit the engineering characteristic the closest. It needed to have a clean appearance and be able to stand on its own, which the vertical option may not have been able to do. Therefore, the horizontal option is the chosen design.

Table 2: Physical Design Decision

Criteria	Weight	Options			
		Horizontal		Vertical	
		Score	Total	Score	Total
Cost	5	5	25	5	25
Ease of Implementation	4	4	16	3	12
Aesthetic	3	5	15	2	6
Total:			56		43

Table 3: Software Design Decision

Criteria	Weight	Options	
		OpenAI Gym	Personal Implementation
		Score	Total	Score	Total
Ease of Implementation	4	5	20	2	8
Open Source	3	5	15	0	0
Total:			35		8

Table 4: File Format Design Decision

Criteria	Weight	Options					
		Custom File Format		DDR Format		Guitar Hero Format	
		Score	Total	Score	Total	Score	Total
Cost	5	2	10	4.5	22.5	3	15
Ease of Implementation	4	3	12	4	16	2	8
Open Source	3	1	3	4	12	2	6
Total:			25		50.5		29

Table 5: DANNA2 Implementation Decision

Criteria	Weight	Options			
		Simulation		FPGA	
		Score	Total	Score	Total
Cost	5	5	25	1	5
Ease of Implementation	4	4	16	2	8
Open Source	3	4	12	1	3
Total:			53		16

Overall, the design decisions we made in this project are based on three criteria: financial cost, ease of implementation, and open source accessibility. To ensure that our decisions held true to these criteria, we utilized weighted decision matrices (Tables 2 - 5) with a five-point scale to evaluate them. The use of these criteria are confirmed for the decisions to base the game environment on OpenAI's Gym framework and the choice to use a modified version of the *Dance Dance Revolution* step map file format. These decisions alone make this project more digestible and more accessible to other developers than it would be via the other paths discussed previously. The choice to use a neuromorphic simulation over an FPGA also follows these criteria as the centralization of the system reduces complexity -- and increases accessibility accordingly -- reduces cost, and makes for a simpler integration with the game environment. While the physical design decisions do not have much bearing on open source accessibility, they do affect cost, and each poses its own implementation challenges. As can be seen by the weighted decision matrices, every design decision made is, by our metrics, the optimal decision.

V. Embodiment Design

The modules that constitute the software portion of RATMANN consist of the rhythm game environment, the wrapper environment born from neuromorphic network training requirements, and the visualization program. The

rhythm game environment essentially drives a round of a rhythm game. It has configurable parameters such as the track length, note speed, and score thresholds, which are set via JSON files passed in on declaration. Also passed in on declaration are a filename for a modified version of a step map file and song difficulty. A step map file is the file format used by the popular rhythm game *Dance Dance Revolution*, and it contains all of the necessary information to allow the environment to represent the song such as BPM and all of the notes that the user must play. These files are modified to include a fifth input on each note so as to utilize all five buttons on RATMANN's cabinet. The environment uses this information and the notes to produce an array of all of the notes for every difficulty in the file and calculates the time steps at which BPM changes and stops will occur, if they are specified in the step map file. Since the speed as specified in the aforementioned parameter file is held constant, the spacing between the notes must vary in order to properly display the notes. This is accomplished on a per-measure basis with the following equation:

$$s = \frac{v * 240}{n * bpm * dt}$$

where s denotes the note spacing, v denotes the speed of the notes, n denotes the number of notes in the currently handled measure, bpm denotes the current BPM, and dt denotes the period at which the environment is stepped through.

The environment steps through the song by means of the step member function. This function takes as parameters the actions taken by whatever agent is playing the game, and it returns the resultant state of the game, the reward earned for that particular action, whether or not the game is completed in the form of a boolean, and any extraneous information. The body of this function progresses the state of the game based on the given action -- the standard pattern of operation for an OpenAI Gym environment. Specifically, this means that the action is compared to whatever notes are playable, if any, and the score is calculated based on how close the note is to the end of the play area. The notes are driven forward based on the note speed and spaced out according to the equation mentioned previously, and they are either removed from the visible area or added to it based on their calculated positions. Using the spacing equation, BPM information, note speed, and the number of notes in the current measure, the environment tracks the position of the next note to be added to the track. If a note is in the visible range, it -- along with its distance from the end of the track -- is added to an array. This assists the visualization of the game and simplifies producing state information. The array essentially functions as a queue where the first note and its distance are used as the state information that is returned in the step function. Currently, the only condition that ends the game is that the environment runs out of notes to play. A reset function is also included in the environment for convenience that simply sets an iterator that tracks the current measure to zero and resets some other important state variables.

For the purpose of training and interfacing with neuromorphic networks, an instance of one of these game environments is enclosed in a wrapper environment. The wrapper environment alters the input from the network and the output from the original environment to be more agreeable with EONS, the evolutionary approach that is used by UT's neuromorphic group to train neuromorphic networks for specialized tasks. EONS begins with a population of randomly generated neuromorphic networks. It then observes how well each network performs the given task and notes the top-performing networks. It then grafts parts of these networks together to form children. EONS tests these children as it did the original population, and the process is repeated for a specified number of epochs. Upon attempting to train with the original environment, it was determined that the scale of the state data was simply too large for EONS to learn in a reasonable amount of time. Thus, this wrapper environment only has one parameter as input and one parameter as output. The output parameter is either 0 or 1 depending on how close the note of the interior environment is to the end of the play area. This threshold is determined by a network efficacy parameter. The input parameter is a "play" signal; when the wrapper environment receives a 1, it plays the most recent note that it observed from the interior environment, and it does nothing otherwise. The score and game status returned are identical to the interior environment. On top of leading to reasonable network training times, this approach allowed the trained network to be transferable between songs. This means that the training process did not have to be repeated for every song included in RATMANN, which is quite convenient. A user-controlled instance of the rhythm game environment and a network-controlled wrapper environment are run side by side within the project's visualization module.

For the visualization of the game, we chose to use the Pyglet library for Python. Utilizing this library, we were able to create a main menu, the play screen, and restart menu. The Pyglet library also allowed for testing the game using keyboard presses before we continued onto testing with the hardware. The flow of the game begins with the main menu, where the player can select between five song choices. The songs were chosen based on different difficulty levels and how closely they fit the theme of our game. Once a song has been selected, the music begins to play and the notes appear on the screen. The notes are depicted as cheese wedges, and there are icons on the bottom of the screen that indicate where the notes need to be pressed. The icons are transparent and light up when pressed, either by a keyboard press or by a hardware button press. The score changes based on when the button is pressed. The score goes down by negative one if the button was pressed, but no note was in the threshold area. Depending on how close the note was to the end of the track, the score will go up by either one, two, or three. At the end of the game, the player will receive a display of one, two, or three stars

depending on how well they performed on that song. The player will be able to see the score of the neural network in addition to their score. The player then can choose to repeat the same song or choose another song. In the hardware implementation, the player chooses the song corresponding to the color indicated on the screen that matches the color of the button.

Figure 4: RAT MANN Start-Up Game Screen.

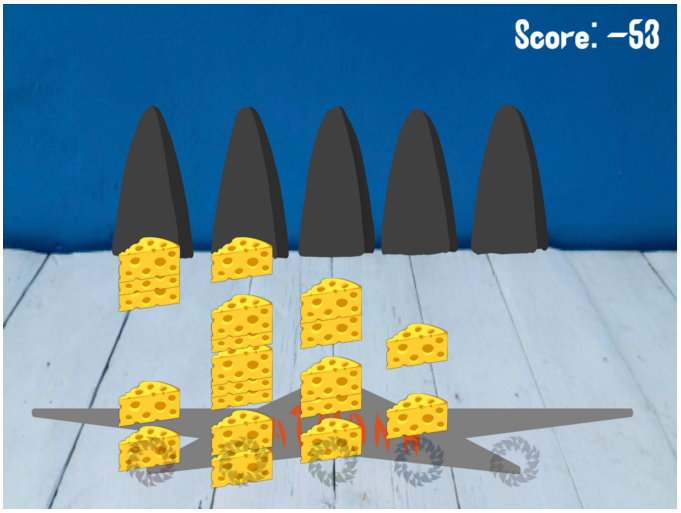

Figure 5: RAT MANN Gameplay Screen.

All the software is hosted with a Raspberry Pi 4 as well as interacting with the hardware that interacts with the physical environment. The Pi interacts with the "player" buttons (as active push buttons and lighting up to indicate use), with the gaming application, and with OpenAI Gym.

The code to interpret button presses is written in Python using the RPi GPIO library that is included on the Pi. This code is essentially a library of functions that are invoked as needed by the application. These functions essentially read the button input and return a number indicating whether it has been pressed or not, and additionally light up the button LEDs correspondingly. The Pi handles running the graphics code as normal as it is loaded with a distribution of the Linux operating system.

To make the arcade system work cohesively, several scripts were written. The functionality of these scripts is to run the application code on startup and to correctly size and position the windows on the screens.

The build of the arcade-style housing can be seen in Figure 6 and a close up of the internal setup of the electrical components can be seen in Figure 7.

Figure 6: Arcade-Style Housing and Internal Supports.

Figure 7: Electrical Component Placement.

VI. Test Plan

Several of the Engineering Characteristics determined are directly related to the implementation of the project. These therefore can be tested by evaluating if we successfully implemented these characteristics.

Number of Players

The number of players was defined to be two, where one player is the neuromorphic player, and the other is the human player. The success of this characteristic can be determined by the success of its two dependent characteristics: the playable app, and the OpenAI Gym interface.

Microcontroller

Our choice of microcontroller was a Raspberry Pi. We previously had considered utilizing an arduino in tandem with a Pi; however, after evaluating our budget and scope, it was

determined that the benefit and ease of use of using both controllers would not outweigh the other areas we would have to cut back in order to stay within the budget. Our metric of success for the microcontroller is whether or not it is able to complete all of our other functions.

Input

The input for this device was determined to be an array of buttons for each player - five for the neuromorphic player, and five for the human player. Our finished product includes five buttons per player and has been successfully integrated into the rest of the design.

Output

The output for this device was determined to be two screens and audio. We were able to successfully integrate two screens and audio capability with our microcontroller of choice.

Neuromorphic Model

Our sponsor required that our project be compatible with DANNA2 as well as a newly created neuromorphic model called GNP. Since both of these network models are usable with OpenAI Gym, our metric of success for this characteristic will be the OpenAI Gym interface.

OpenAI Gym Interface

OpenAI Gym was selected as the environment to use for application development because of its compatibility with the needs of our sponsor as well as its status as an open source tool. Some of the experiments we can use to determine the efficacy of our implementation are as follows:

1. Use of a "brute force" model to ensure that the environment functions as intended,
2. Application can be played by TENNLab Neuromorphic models, and efficiency scores can be generated
3. Application can be played and scored with another open source model compatible with OpenAI Gym

The results of our experiments were that all were successful and functioned as expected. The test matrix for this test is included in the appendix.

Playable App

The efficacy of our implementation of the *Guitar Hero* and *Dance Dance Revolution* inspired game can be determined through several tests:

1. Application generates a sequence of notes along with a song
2. Application receives input (from keyboard or buttons)
3. Application generates a score based on actual input combined versus expected input
4. Application counts points for button presses that occurred within the time threshold
5. Application and graphics runs smoothly on the Raspberry Pi

The results of our experiment were that our implementation passes 1-4 without question. A song input to the game generates a sequence on the screen and the application successfully parses input and generates a score. The only experiment that encountered difficulty was experiment 5. The initial issue encountered was that as a button on the full arcade system was being pressed, the visualizer for the game was lagging. This was because the button read and button light-up logic included a short delay, but the delay occurring after every read was adding up to produce a visual lag. This was fixed by slightly modifying the logic to interpret a button press. The test matrix for this test is included in the appendix.

VII. Deliverables

There are five main deliverables, which are listed below:

- OpenAI Gym/Model Interface
- Guitar Hero OpenAI Gym Application
- Electrical Circuitry
- Firmware Code
- "Arcade-style" Housing

The main deliverable for this project will be the *Guitar Hero* OpenAI Gym application. This application will allow a player and/or AI to play it, and produce a score based on their performance on the chosen song. The application and the OpenAI Gym interface will be written in Python. Furthermore, the game will have been thoroughly tested for any bugs by the end of the Spring Semester 2021. There will be electrical circuitry that will be utilized to display the game and allow a user to play the game. The electrical circuitry will include a power converter for wall power use, buttons for each player, a Raspberry Pi, wiring and harness connections, speakers, and two LCD screens. The firmware code will be written for use on the Raspberry Pi and it will allow communication between the hardware and the application code. Finally, "Arcade-style" housing will be created to contain the design and give it a polished final-product appearance. This housing will also protect the parts from accidental damage.

VIII. Project Management

The team developed the following schedule to ensure that design requirements and engineering characteristics for the project are met for the duration of senior design (i.e. both the fall and spring semesters). The milestone timeline also allows for implementation of our stretch goals in addition to our base requirements.

Table 6: Project Milestones - Original

Date	Milestone
1-Oct-20	Define the project and engineering characteristics

15-Nov-20	Budget for hardware determined, game environment created
15-Dec-20	Initial hardware build, initial housing build, initial firmware complete
15-Jan-21	Gaming application complete and able to train networks
15-Feb-21	Hardware testing and validation, testing with firmware, start building interfaces between hardware, firmware, software
1-Mar-21	Housing for hardware completed, first attempt at combining all components complete
1-Apr-21	Final changes to hardware and housing complete, final changes to full system complete, begin bug fixes and stretch goals
15-Apr-21	Project completion (senior design requirement)

This project heavily relies on the implementation of a neuromorphic model, creation of a gaming interface, and visualization of the neuromorphic fitness levels; this project therefore relies heavily on three computer engineering students to form the software team (Jaidin Jackson, Megan Stanton, Alex Twilla). The single electrical engineering student (Jessica Stevens) is the sole member of the electrical team and is to determine the electrical hardware needed for implementation and will also design the "arcade-style" housing for the completed presentation of the project. Finally, the remaining computer engineering student (Samantha Zimmermann) is the sole member of the firmware team and is in charge of the firmware development required to bridge the gap between the electrical hardware and software components of the project.

Throughout the course of this project, several changes and issues came up throughout that delayed our milestone progress. Some of the issue was a misunderstanding of how development will go. For example, we thought developing the application as well as training networks to play it would be a simultaneous process; upon actually beginning the project, it became apparent that training the neural network to play the game must be done after the application is complete. Additionally, our sponsor later requested that we write complete documentation so that the project can be later reused or repurposed by someone else in the research group. Below is an updated milestone table that reflects when each milestone was actually completed.

Table 7: Project Milestones - Updated

Date	Milestone
1-Oct-20	Define the project and engineering characteristics
15-Nov-20	Budget for hardware determined, game environment created
15-Dec-20	Initial hardware build, initial housing build, initial firmware complete
15-Jan-21	Gaming application complete
15-Jan-21	Firmware code complete
15-Jan-21	Final changes to housing complete
20-March-21	Hardware testing and validation, testing with firmware, on
	partially assembled design. Start building interfaces between hardware, firmware, software
14-Apr-21	Networks able to be trained and play the game
20-Apr-21	Hardware circuit fully assembled
24-Apr-21	Full system integration firmware code complete, begin debugging and testing
26-Apr-21	Anticipated project completion (senior design requirement)
4-May-21	Complete documentation

IX. Budget

Table 7 and Table depicts the two budgets used to fulfill the design requirements as specified for the project. If more than one was purchased, the quantity is listed in parenthesis. However, during implementation we encountered some issues and had to get some alternatives to perform the same function. Table 7 was the original budget with the items ultimately replaced in red. Table 8 is the updated budget with the replacement items in green. You can see these are mostly power components and computing storage which was originally overlooked. This could ultimately result in an expansion of the project idea in a future senior design project.

Table 8: Original Budget

Category	Part	Price
Electrical		
Computing	Raspberry Pi 4 model B, 4GB	$55.00
Cables	Micro HDMI to HDMI Cable (2)	$17.90
Display	Pimoroni HDMI 10" LCD Screen Kit (2)	$279.50
Audio Related	3" Diameter Speaker, 4 Ohm 3 Watt (2)	$3.90
	Stereo Audio Amplifier, MAX98306	$8.95
Buttons	LED Arcade Buttons (10)	$25.00
Power Supply	AC/DC Converter, 5V 40W	$17.97
	Toggle Power Switch	$4.02
	4 Connector Power Rail	$1.70
	AC/wall power cord	$6.75
	Cable Gland	$3.60
Mechanical		
Mechanical Hardware	4-40 button head hex drive screws	$11.34
	4-40 low strength steel hex nuts	$0.89
	Heavy Duty Rubber Bumpers	$15.24
	M2.5 Button Head Hex Drive Screw	$8.46
	Brass M2.5 Standoffs Black Plated	$6.25
Building Materials	1/8"x2'x4' Tempered hardboard (2)	$7.96
	High-Strength Gorilla Glue, 4 oz	$8.28
	Clear Glass	$5.34
	Blacktop Black Core Matboard	$11.89
	Total	**$499.94**

Table 9: Updated Budget

Category	Part	Price
Electrical		
Computing	Raspberry Pi 4 model B, 4GB	$55.00
	64GB Samsung SD Card	$16.00
Cables	Micro HDMI to HDMI Cable (2)	$17.90
Display	Pimoroni HDMI 10" LCD Screen Kit (2)	$279.50
Audio Related	3" Diameter Speaker, 4 Ohm 3 Watt (2)	$3.90
	Stereo Audio Amplifier, MAX98306	$8.95
Buttons	LED Arcade Buttons (10)	$25.00
Power Supply	Three outlet extension cord	$3.47
	5V wall adapter (3)	$3.00
	Micro-B USB to USB cables (2)	$6.00
	USB-C USB to USB cables (1)	$5.00
Mechanical		
Mechanical Hardware	4-40 button head hex drive screws	$11.34
	4-40 low strength steel hex nuts	$0.89
	Heavy Duty Rubber Bumpers	$15.24
	M2.5 Button Head Hex Drive Screw	$8.46
	Brass M2.5 Standoffs Black Plated	$6.25
Building Materials	1/8"x2'x4' Tempered hardboard (2)	$7.96
	High-Strength Gorilla Glue, 4 oz	$8.28
	Clear Glass	$5.34
	Blacktop Black Core Matboard	$11.89
	Total	**$499.37**

X. References

[1] “01278.70.01 General Cable/Carol Brand: Cable Assemblies,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/general-cable-carol-brand/01278-70-01/2758008. [Accessed: 10-Nov-2020].

[2] “1314 Adafruit Industries LLC: Audio Products,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/adafruit-industries-llc/1314/7902282?s=N4IgjCBcoLQdIDGUAuAnArgUwDQgPZQDa4ArAEwAcIAugL517nHgDMYALLXUA. [Accessed: 10-Nov-2020].

[3] “1546306-4 TE Connectivity AMP Connectors: Connectors, Interconnects,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/te-connectivity-amp-connectors/1546306-4/1277415. [Accessed: 10-Nov-2020].

[4] “1799 Adafruit Industries LLC: Connectors, Interconnects,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/1799/1528-2453-ND/7241409?itemSeq=343198098. [Accessed: 10-Nov-2020].

[5] “3487 Adafruit Industries LLC: Switches,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/adafruit-industries-llc/3487/7364334?s=N4IgjCBcoLQdIDGUAuAnArgUwDQgPZQDa4ArAEwAcIAugL517nEgDMALJQOy11A. [Accessed: 10-Nov-2020].

[6] “3489 Adafruit Industries LLC: Switches,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/adafruit-industries-llc/3489/7349495?s=N4IgjCBcoLQdIDGUAuAnArgUwDQgPZQDa4ArAEwAcIAugL517nEgDMALJQJy11A. [Accessed: 10-Nov-2020].

[7] “987 Adafruit Industries LLC: Development Boards, Kits, Programmers,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/adafruit-industries-llc/987/5629428?s=N4IgjCBcoLQdIDGUAuAnArgUwDQgPZQDa4ArAEwAcIAugL517nEgCclA7LXUA. [Accessed: 10-Nov-2020].

[8] A. Industries, “Micro HDMI to HDMI Cable - 2 meter,” *adafruit industries blog RSS*. [Online]. Available: https://www.adafruit.com/product/1322. [Accessed: 09-Nov-2020].

[9] A. Industries, “Pimoroni HDMI 10’ IPS LCD Screen Kit - 1024x768,” *adafruit industries blog RSS*. [Online]. Available: https://www.adafruit.com/product/4337. [Accessed: 09-Nov-2020].

[10] A. Industries, “Raspberry Pi 4 Model B - 1 GB RAM,” *adafruit industries blog RSS*. [Online]. Available: https://www.adafruit.com/product/4295. [Accessed: 09-Nov-2020].

[11] “Alloy Steel Button Head Hex Drive Screws,” *McMaster-Carr*. [Online]. Available: https://www.mcmaster.com/button-head-cap-screws/alloy-steel-button-head-hex-drive-screws/. [Accessed: 09-Nov-2020].

[12] Arun, J. Burton, David, Braden, Anders, S. Kannan, Lloyd, A. Gray, Marty, Frank, Thomas, Tom, Jamie, SH2P, Don, Matt, Joss, Chrl93, Kyle, Thibsert, Pd, Nathaniel, Reikred, Moshe, Alex, R. Vaidya, Minmin, PedroKV, Kasbesh, T, Pablo, Lukas, Damien, and F. User, “Raspberry Pi 4-pole Audio/Video Jack,” *Raspberry Pi Spy*, 01-Sep-2020. [Online]. Available: https://www.raspberrypi-spy.co.uk/2014/07/raspberry-pi-model-b-3-5mm-audiovideo-jack/. [Accessed: 10-Nov-2020].

[13] B. Earl, “Stereo 3.7W Class D Audio Amplifier,” *Adafruit Learning System*. [Online]. Available: https://learn.adafruit.com/stereo-3-7w-class-d-audio-amplifier/inputs-and-outputs. [Accessed: 10-Nov-2020].

[14] “Button Head Hex Drive Screw,” *McMaster-Carr*. [Online]. Available: https://www.mcmaster.com/91239A754/. [Accessed: 09-Nov-2020].

[15] C. Editor, “Headphone Jack and Plugs: Everything You Need to Know,” *Headphonesty*, 09-Feb-2020. [Online]. Available:

https://www.headphonesty.com/2019/04/headphone-jacks-plugs-explained/. [Accessed: 10-Nov-2020].

[16] “Compact Plastic Submersible Cord Grip,” *McMaster-Carr*. [Online]. Available: https://www.mcmaster.com/69915K64/. [Accessed: 09-Nov-2020].

[17] “Dimensions 1/8 in. x 2 ft. x 4 ft. Tempered Hardboard (Actual: 0.115 in. x 23.75 in. x 47.75 in.)-109112,” *The Home Depot*. [Online]. Available: https://www.homedepot.com/p/1-8-in-x-2-ft-x-4-ft-Tempered-Hardboard-Actual-0-115-in-x-23-75-in-x-47-75-in-109112/202585358. [Accessed: 09-Nov-2020].

[18] “Gardner Glass Products 16-in x 20-in Clear Glass,” *Lowes*. [Online]. Available: https://www.lowes.com/pd/Gardner-Glass-Products-16-in-x-20-in-Clear-Glass/3121143. [Accessed: 09-Nov-2020].

[19] “GPIO,” *GPIO - Raspberry Pi Documentation*. [Online]. Available: https://www.raspberrypi.org/documentation/usage/gpio/. [Accessed: 10-Nov-2020].

[20] “GTS447A101HR CW Industries: Switches,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/cw-industries/GTS447A101HR/3193576. [Accessed: 10-Nov-2020].

[21] “High-Strength Gorilla Glue, 4 oz. Bottle,” *McMaster-Carr*. [Online]. Available: https://www.mcmaster.com/7454A22/. [Accessed: 09-Nov-2020].

[22] “IRM-45-5ST MEAN WELL USA Inc.: Power Supplies - External/Internal (Off-Board),” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/mean-well-usa-inc/IRM-45-5ST/7704687?+IP4BbALTUkrEADZk8gCbdlYAAwR2XSCBDEOATzb5umXGilA. [Accessed: 10-Nov-2020].

[23] “Low-Strength Steel Hex Nut,” *McMaster-Carr*. [Online]. Available: https://www.mcmaster.com/90480A005/. [Accessed: 09-Nov-2020].

[24] MilliwaysMilliways 45.2k1919 gold badges7676 silver badges147147 bronze badges and joanjoan 59.8k44 gold badges5656 silver badges8989 bronze badges, “Raspberry Pi Power Limitations,” *Raspberry Pi Stack Exchange*, 01-Oct-1965. [Online]. Available: https://raspberrypi.stackexchange.com/questions/51615/raspberry-pi-power-limitations. [Accessed: 10-Nov-2020].

[25] R. Brothers, “Arcade Button Control Box,” *Adafruit Learning System*. [Online]. Available: https://learn.adafruit.com/arcade-button-control-box/overview. [Accessed: 10-Nov-2020].

[26] “Unthreaded Bumper,” *McMaster-Carr*. [Online]. Available: https://www.mcmaster.com/9540K42/. [Accessed: 09-Nov-2020].

XI. Appendices

GitHub Link: https://github.com/mstanto4/SeniorDesign

Business Model Canvas

Designorate.com

Company Name:	Date:
RAT MANN	October 18, 2020

Key Partners	Key Activities	Value Proposition	Customer Relationships	Customer Segments
- TENNLab group and sponsor. - Engineering team. - Senior Design professor. - Indivudals involved in distribution.	- Find cost effective means to distribute the units. - Network with other institutions to bring awareness and value of product. - Create a means for consumers to troubleshoot issues. - Continual development.	- Creating a digestible demo of AI/ML. - Recruitment into the AI/ML field. - Open source.	- Supplying the product to TENNLab who coordinates with other institutions.	- Secondary and post-secondary institutions.
	Key Resources		**Channels**	
	- TENNLab's neuromorphic model(s). - Utilize OpenAI Gym. - UT's and TENNLab's connections to other institutions. - Use GitHub issue tracking to communicate with consumers.		- TENNLab to start. - Efficient due to TENNLab's connections to high schools and other universities.	

Cost Structure	Revenues Streams
- 80% costs from hardware. - 20% costs from housing materials. - Development done for free for open source software.	- Sales to secondary and post-secondary institutions.

OpenAI Gym Interface Test Matrix	
Test	Experiment Results
Use of a “brute force” model to ensure that the environment functions as intended	Successful
Application can be played by TENNLab Neuromorphic models, and efficiency scores can be generated	Successful
Application can be played and scored with another open source model compatible with OpenAI Gym	Successful

Playable App Test Matrix	
Test	Experiment Results
Application generates a sequence of notes along with a song	Successful
Application receives input (from keyboard or buttons)	Successful
Application generates a score based on actual input combined versus expected input	Successful
Application counts points for button presses that occurred within the time threshold	Successful
Application and graphics run smoothly on the Raspberry Pi	Needs work