Applications in Artificial Intelligence: Neural Networks in Verilog, Optical Character Recognition, and a Maze Solving Robot Based on Q-Learning

***Abstract* – This paper concludes the two-semester project by explaining this team's societal problems, design idea contract, funding, and milestones. It will also discuss the work breakdown structure of this semester's work, risk assessment, design philosophy, deadline restrictions, design decisions, practical challenges, and achievements.**

Key Terms – Artificial Neural Network (ANN), Optical Character Recognition (OCR), Field Programmable Gate Array (FPGA), Artificial Intelligence (AI), Activation Function, Micromouse

Breeana D. Proffit
Computer Engineering
California State University,
Sacramento, USA
proffitbreeana@gmail.com

Sudhakar Alla
Computer Engineering
California State University,
Sacramento, USA
sudhakaralla1@gmail.com

Vikram Saroay
Computer Engineering
California State University,
Sacramento, USA
vikramsaroay@gmail.com

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

EXECUTIVE SUMMARY

Over the course of ten months of Senior Design Project at California State University of Sacramento, this team has explored three avenues of artificial intelligence. Compared to other groups in this team's graduating class, this team has taken a broad approach to its project design. This is, in large part, due to unpredictable complications associated with pursuing several topics the members of this team were initially entirely unexposed to only to realize that the time frame provided by the class – as well as the budget available to this team – made more in-depth research and development of two of the initial project ideas infeasible. This team pursued the ideas to the best if its abilities given the resources it did have.

During the end of summer and most of the fall 2017 semester, the focus was on the Verilog design. It was approaching its penultimate design when the issue of the project's growing complexity requiring more space than was provided by any reasonably attainable FPGA swiftly brought that portion of the project to its end.

For the latter half of the same fall semester, this team developed the concept, design, and documentation for an optical character recognition system using a neural network similar to the one this team had designed earlier in the semester in Verilog. The software design initially began as a software simulation to compare to the same processes this team was attempting to create in Verilog.

Then, for winter break and the spring 2018 semester this team switched its focus to a robotic system that would make use of a Q-learning algorithm to solve mazes. The reason for this switch was pragmatic – the pursuit of a high-speed optical character recognition that could determine entire words was infeasible given the difficulty presented in just recognizing single characters. The use of reinforcement learning was in line with our prior pursuits.

Each of the three aspects of this project required extensive research, and in summary the members of this team have all developed a comfortable understanding of: the function and design of artificial intelligence systems; the difference between supervised and reinforcement learning algorithms; image processing to extract edges and other characteristic features of handwritten characters for use in neural networks; organization modules and the design of several floating-point features in Verilog; the entire robot design process, including balancing the factors of weight, motor torque, battery capacity, and model size; the use of feedback systems such as PID controllers for ensuring precise movement in robotic systems; and several other skills.

I. INTRODUCTION

This paper describes in depth every aspect of this team's project, focusing primarily on the work done during the fall semester with Optical Character Recognition (OCR) and the spring semester with the Hyper Expedient Robotic Maze Extraction System (HERMES). The paper summarizes details about the two societal problem that this team attempted to solve and covers the design idea contracts and changes made to them. Additionally, this paper will provide an overview of funding, milestones, work breakdown structures, risk assessments, and design choices. To finish, this paper will describe both prototype's status at the end of their respective semesters, and how these products may have a place in a consumer market.

II. SOCIETAL PROBLEM

During the fall semester, this team focused on finding a solution to aid dyslexic students. Dyslexia is a worldwide societal problem that prevents young children from focusing on their studies and limits adults from doing certain tasks. A very common form of dyslexia that exists worldwide is known as visual dyslexia, where an individual has a difficult time reading words/phrases. Mental disorders such as visual dyslexia have no permanent cure as stated by most healthcare facilities [1].

There are, however, many treatments that aid individuals for a good amount of time so that they can adapt to the situation. For example, some dyslexic children are enrolled with teachers/mentors, one to one, and are educated with specific needs to their problems. Our team's plan is to create an ANN system that is able to perform Optical Character Recognition (OCR) and would make it easier for dyslexic individuals to overcome learning obstacles.

An OCR system would allow dyslexic students to correctly identify letters and numbers so that they can piece together an understanding of which characters are affecting their situation and act/adapt accordingly.

Classical OCR systems were designed in software and used a lot of the energy of the processor since they were not run in parallel. Running the OCR system on a portable hardware using ANN is ideal when it comes to solve the problem of converting handwritten or digital characters into speech.

During the spring semester, this team's focus shifted to introducing students to the concepts of robotics and artificial intelligence. As American math and science literacy levels fall, it has become evident that the education system is in need of help. One of the most prestigious nationwide tests is the Programme for International Student Assessment (PISA), which every

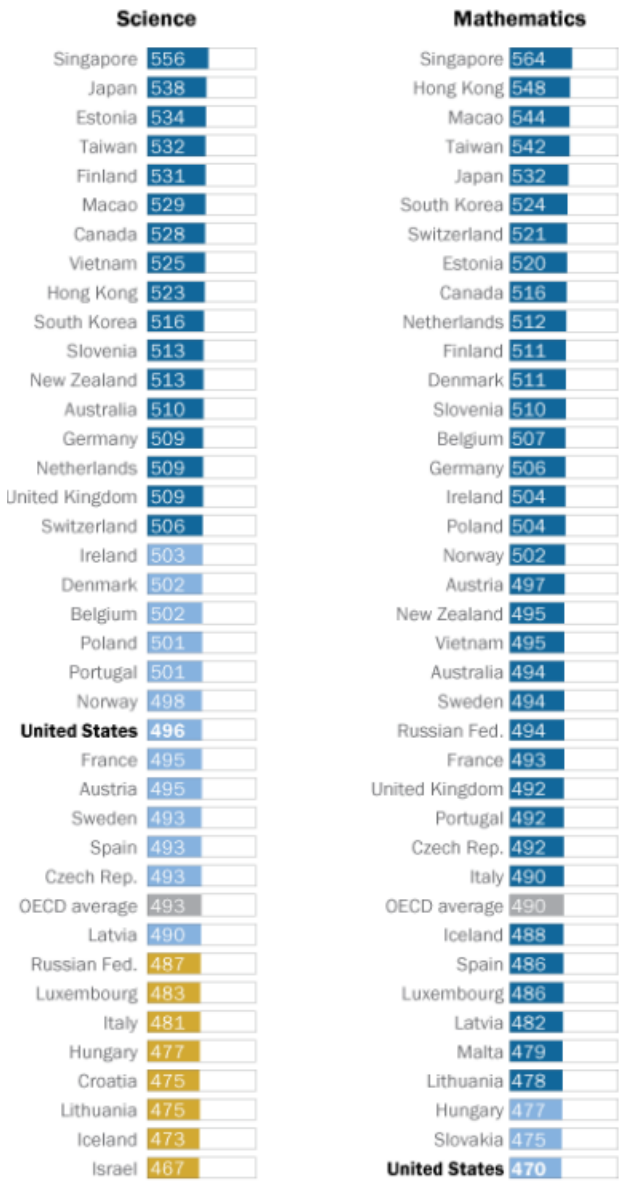


Figure 1. Rankings in Science and Math [2]

three years measures reading ability, math, science, and other key skills among fifteen-year-olds in dozens of developed and developing countries. The most recent PISA results from 2015 place the USA at an unimpressive 38th out of 71 countries in math and 24th in science literacy. [2].

III. DESIGN IDEA CONTRACT

This team's exploration into various Artificial Applications has spanned three different projects and two semesters, lending itself to a myriad of various project milestones. This project's Design Idea Contract for the spring 2018 semester is different from that proposed in the fall 2017 semester for the simple reason that a significant portion of this project was altered in response to unforeseen challenges arising from designs in the fall semester. Whereas the 2017 contract focused on the design of an artificial neural network that could act as an optical character recognition system, the 2018 contract focuses on HERMES, the maze-solving robot.

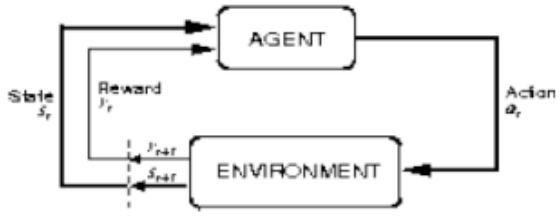


Figure 2. Agent-Environment Relationship [4]

Figure 3. Micromouse Competition Maze [5]

A. 2018 Design Idea Contract

The design idea contract can be well described in the context of *Figure 2* and Figure 3. The HERMES robot design is comprised of three primary features: Mind, Body, and Sense.

The Mind feature is made up of Q-learning, which is a learning algorithm using an agent-environment relationship as described in *Figure 2.* In this project, the environment is the world itself provided to the agent through sensor data, while the agent is the algorithm is what decides which direction to take next. For the Mind feature, the design contract requires a full implementation of Q-learning, maze solving capability, and efficiency feature upgrades.

The Body feature consists of motors, control and feedback systems, and chassis design. The focus with this feature is providing precise movement to assure that the Mind feature is always roughly correct about its understanding of where it is in the real world. The specific features required by the design contract are straight horizontal movement and clean ninety degree turns.

The Sense feature is made up of sensors and what is accomplished with sensor data. This provides an understanding of obstacles and variance from desired angle and position. The specific features for the Sense feature are wall detection, wall avoidance, and end detection which is essentially making stops if the robot approaches a wall too close.

These features are tested in and out of a maze-like environment. A maze such as that shown in *Figure 3* is used to test the features listed above working together, but each feature is tested individually as well to ensure that this team has met the design contract's specification for every feature.

B. 2017 Design Idea Contract

To briefly discuss the ideas covered in the 2017 design contract, this team's primary goal for that semester was to develop an optical character recognition system which would accept a real-world written character as input and output the character as recognized by the system.

Originally this contract had a lean toward the use of an FPGA-based neural network for use in optical character recognition, but as issues arose the focus shifted toward generally optical character recognition with a neural network of our own design running a microprocessor.

The specific features required by that contract consisted of an implementation training datasets to train a neural network, the creation of a multilayer neural network, an implementation of training for that neural network, the ability for the system to be trained in other topics, and a user-friendly interface.

IV. PROJECT MILESTONES

The milestones accomplished within this project are categorized neatly into the three project ideas this team pursued. The milestones pertaining to those three ideas are given in subcategories for neatness.

A. FPGA-Based ANN

Major milestones for this particular project idea consisted of the development of working, generalized Verilog modules used in the larger top-level design. An example of a neural network is given in *Figure 4*.

The first such milestone came with the development of floating point multiplication and addition modules. The back-propagation model made use of floating point numbers in all aspects of the system – in synapse weights, in input summation, in neuron's the activation function, and in back-propagation modules. Inputs and outputs to the system are also floating-point values.

The next milestone for this FPGA design was the development of a sigmoid activation function approximation module. In a neuron model, an activation function is used to transform a summation of input values to an output between 0 and 1 in a non-linear manner. The activation function used for the FPGA model is a sigmoid function, which is described in the equation below.

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}$$

This function outputs a curve like the one shown in *Figure 4*, described there as "Hyperbolic Tangent Sigmoid."

The activation function was particularly difficult to achieve in Verilog due to its non-linear nature. A direct representation is possible but highly complex, so the milestone was accomplished through the use of a minimax approximation.

The third milestone accomplished was the development of a neuron, which consisted of weight multiplication, input summation, and an activation function working in sequence.

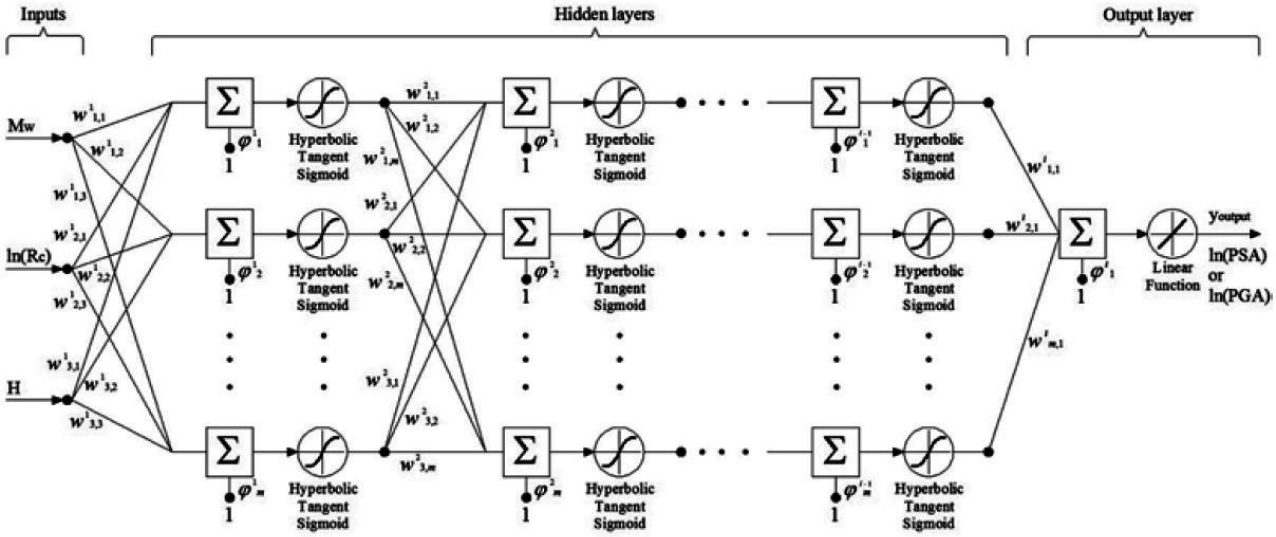


Figure 4. Artificial Neural Network [6]

The fourth milestone for the FPGA model was the development of a neural network of five neurons, connected from input to output. This was difficult primarily as a result having to debug the system with floating point values, but it ultimately the process of translating floating point to decimal then determining what issues existed to make floating point values go astray was developed enough to allow this team to reach this milestone.

And the fifth milestone accomplished was the creation of back propagation modules to connect output values to every layer of neurons. Back propagation equations for each weight were derived by hand and generalized for up to three layers so as to make adding more neurons easier in the future, and those equations were translated to Verilog modules.

B. Optical Character Recognition ANN

During the second half of the fall semester, this team began to work on a software version of the OCR system that was being developed on an FPGA. To accomplish the goals of the 2017 design contract, the software OCR system followed the same general guidelines and resulted in the following milestones.

The first major milestone this subproject achieved was being able to take pictures using a Raspberry Pi and converting them to a grayscale 16x16 matrix. This milestone involved using TTL serial cable camera made for the Raspberry PI. The camera module provided a 1920x1080 image which surpassed the desired size restrictions of the neural net. To bring it in line with the 16x16 pixel size used by the neural network, this team used the Python Image Library (PIL) and its resizing functions. Additional functions provided by PIL allowed the system to grayscale the image and then convert it to a 16x16 matrix.

The next major milestone was developing a C++ based ANN that would be able to use the 16x16 matrix generated using the RPI camera and PIL. This team strictly followed the restrictions provided by for the FPGA model in order to accurately represent the same kind of network. These restrictions dictated the learning system used (in this case Back Propagation), the applied learning rate, and the activation function (in this case the Sigmoid function), and more.

Once the neural network system was developed, it had to be trained. The first iteration of the OCR system used the MNIST library available online which is a compilation of 60,000 handwritten letters and numbers. Training the neural network with this library was yet another milestone.

Verification “live” testing proved to be nearly impossible because the image quality captured by the RPI camera could not match the quality of the scanners used by the MNIST dataset developers, so the next milestone was in creating and training the neural network with a custom dataset was generated using hundreds of handwritten numbers between zero and nine. These handwritten numbers had to be different enough for the system to begin to recognize variances in input, and to provide this variety data was collected by asking colleagues in the CSUS Engineering Labs to write characters for this project.

The number of samples varied, as some numbers such as one (1) and seven (7) proved difficult to differentiate without enough data due to their similarity. Once the custom PNG dataset was developed, it was processed with the tools resulting from the first milestone, and the C++ANN was finally properly trained.

Once the C++ OCR ANN was properly trained, the next milestone was to use the neural network trained by the new custom dataset to accurately recognize characters from pictures taken by the Raspberry Pi camera. This milestone was accomplished and presented at the fall semester Senior Design exhibition.

C. Maze-Solving Robot with Q-Learning

The very first milestone in the spring semester was the understanding and implementation of the Q-learning problem solving approach to the Micromouse problem. This approach included discovering the various mathematics problem solving techniques when it comes to making decisions, for example choosing which direction to go in a maze. Markov’s Decision Process and Temporal Difference are techniques that lead to the fully working Q-learning algorithm. The Q-

learning is the main ingredient of the maze solving algorithm.

The main focus during the spring semester was to implement a full proof custom Q-learning algorithm onto a Micromouse, therefore the first major milestone was the complete development of the mind, or the maze solving capability, coded in python. The software implementation of the algorithm proved that the agent inside the maze environment is able to explore and reach the final destination, no matter the size of the maze. The largest maze tested during iterations was 256 by 256 spaces, which took around 8mins for the agent to explore the maze and reach the end goal, at a very high speed run.

Later the milestone of efficient feature upgrades was achieved in the maze solving algorithm. The issue that kept occurring was that the agent wanted to keep going back to the direction that it already explored in which it found not luck previously. This issue would randomly occur in the iterations, therefore the solution was to give a higher negative reward to the agent when it wanted to visit the older locations that had no positive outcome.

The next milestone was the wall detection capability. The first approach at detecting the walls was using the Sonar distance sensors on V1, V2, and V3 of HERMES. The Sonar sensors required multitasking to operate in parallel, which would slow down or interfere with other operations on the Raspberry Pi.

Therefore, the next best option was to use IR sensors that gave a binary feedback, whether the wall was there or not. This strategy allowed a smooth transition from the Q-learning algorithm to the hardware. The idea was to sense walls around the agent and capture these walls at every step of the maze. Every version after V3 used IR sensors for wall detection.

Wall avoidance was the next big milestone which gave the agent a little more room to be flexible while it is roaming around the maze. The wall avoidance was achieved using the IR sensors which were set at a comfortable distance from the walls on each side. The wall avoidance system also helped with the straight paths.

Another milestone that strengthened the straight horizontal movement was the PID controls using the encoder feedback. V1 to V3 of HERMES used stepper motors, without any sensor feedback, however the precision was very high when going straight. The servo and DC motor versions required a sensor feedback to tell the motors constantly if they are going straight or not.

The encoders were used on V11 of HERMES, in conjunction to the 12v DC motors. The PID controller allowed to correct the error that accumulates due to the difference in speed on each motor. Also the slight right and left shifting of the motors, due to signal noise and bad motors, was fixed by the PID and correct function.

Along with the straight horizontal movement, the 90 degree turn control was achieved. The 90 degree turns were achieved by the IMU feedback, which uses 9 degrees of freedom to calculate the current angle of the robot. The starting position of the robot determines the zero degrees and from there, the robot can determine north, east, west, and south. The IMU was first included in V9 of the build, in conjunction to the servo motors.

The most recent milestone achieved was the front wall detection to make the agent stop in time and make a decision. As the agent approached a wall in the front, it needs enough space to make a quick stop, so it does not collide with the wall, therefore a control strategy is needed for such situations.

The strategy was to make a right turn as a quick judgement that the front wall is approaching. Two front IR sensors were used at different distances to make the judgement for how close to the front wall the agent is.

V. FUNDING

Fall	**Price (per unit)**	**Source**
Raspberry Pi 3	$40 x2	Amazon
Raspberry Pi Camera	$30	Amazon
Plywood	$10	Home Depo
128gb MicroSD	$48	Amazon
Spring	**Price (per unit)**	**Source**
RPi 3	$40	Amazon
Arduino Uno	$15	Amazon
12v Steppers		
12v DC motors	$17 x2	Amazon
Robot chassis kit	$20	Fry's
IR sensors	$10 x2	Amazon
Sonar sensors	$10	Amazon
12v Battery	$20	Amazon
Servo pack	$20	Amazon
AAA holder	$8	Amazon
AAA pack	$10	Amazon
AAA charger	$17	Amazon
Gear pack	$11	Amazon
L298n	$9	Amazon
L298 7A	$16	Amazon
Mount hub 6mm	$14	Amazon
90x10mm wheel	$12	Amazon
Speed sensor	$11 x2	Amazon
IMU 9DOF	$40 x2	Amazon
Balsa wood	$10	R/C hobby
Pololu DC Motor	$20 x2	Pololu
Pololu Encoders	$12 x1	Pololu
Servomotors	$20 x2	Amazon
6v Steppers	$27 x2	Amazon
70x10mm wheel	$10 x2	Amazon
Screws	$5	Amazon
5V Steppers	$15 x2	Amazon
Wires	$13 x2	Amazon
PCB boards	$10	Amazon
Solder Equipment	$25	Amazon

Table 1. Funding for Fall and Spring

VI. WORK BREAKDOWN STRUCTURES

Given that this project consisted of three subprojects, three work breakdown structures were created. The higher order features are discussed in this section – this team is leaving out low order features from this paper largely because the discussion of the large volume of smaller features distracts from the purpose of this paper. Figures for the work breakdown structures are given in the Appendix G. Work Breakdown Structures.

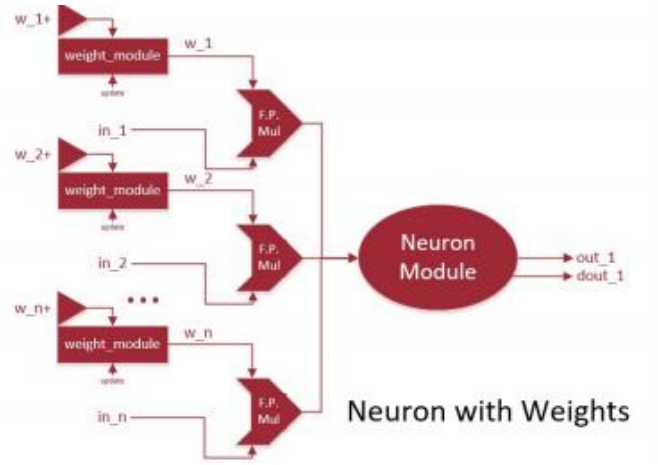


Figure 5. Neuron with Weights

A. *FPGA-Based ANN*

The FPGA Neural Network model, which is as far as this team was able to proceed with this subproject's work breakdown structure, has two primary features – namely it consists of the creation of neurons with weighted inputs as well as the integration of multiple neurons into a network with learning capacity.

Neurons with weighted inputs were comprised of several parts. To begin, the input to the neuron had to be a floating point number which was multiplied by some variable weight – thus the first feature of the neuron was the weight module which performed that function.

After that, weighted inputs were summed together with a tree of floating point addition modules, so the summer tree was the second feature of the neuron.

Then the sum of weighted inputs was sent through an activation function to collapse the output between 0 and 1. The third feature of the neuron was the activation function module which was modeled after a minimax approximation function.

The second part of the subproject's work breakdown structure was the integration of neurons into a network with back propagation.

The neural network was primary made up of a forward-pass portion which is a series of weighted summations and activation functions. This was perhaps the most difficult part of the project to achieve because testing was highly complex and required plenty of calculations and translations between floating point and decimal.

After that, the next portion of the neural network was the development of back propagation modules. These performed all of the necessary work for enabling the neural network to perform supervised learning.

B. Optical Character Recognition ANN

The Optical Character Recognition subproject of the fall semester had three primary tasks – image processing, training dataset creation, and the implementation of an artificial neural network in C++.

The image processing feature allowed this team to format images from the Raspberry Pi camera for the neural network this team used. It resized 1080p images to 16x16 images, one pixel for each input to the neural network.

In order to make the features of characters more obvious in a 16x16 grid, filters were used to enhance edges, to remove color, to increase contrast, and to remove noise with thresholds.

The dataset feature is the set of images the neural network was trained with. These images originally came from the MNIST dataset. This team used the same image processing techniques on the MNIST images as with the camera images, but eventually it was discovered that using that the differences between this team's camera and the camera used to record characters for MNIST was too great. Thus a custom dataset was created for the same purpose as the MNIST dataset.

The final feature is the neural network itself. Beyond the network itself, this feature was broken up into training and testing portions, the latter of which was used to recognize characters written by individuals at the Senior Design Exposition.

The training portion consisted primarily of an interface used to input images of both the MNIST and this team's custom datasets. The neural network was then designed to train using the back propagation learning algorithm.

The testing portion loaded a saved set of trained weights for a neural network into a new neural network. Images were taken, then processed, then sent to the neural network through an interface of this team's design.

C. Maze-Solving Robot with Q-Learning

The three primary features of the HERMES portion of this project were Mind, Body, and Sense.

The Mind portion of Hermes consisted of the Q-learning algorithm, which is a reinforcement learning algorithm used to find the best path to a goal, specifically through a maze in this team's case. This algorithm was enhanced by several efficiency upgrades discussed in the design documentation.

The Body portion of Hermes is made up of the motors, chassis, and battery. These components were switched out several times over the thirteen models this team designed and implemented. The trick to these features was determining the correct ratio of weight, size, torque, and energy capacity.

The Sense portion of this subproject was consisted of all sensors – IMU, digital IR sensors, and encoders – and their interconnection with the other portions of the HERMES project. All controls feedback started with the Sense portion of HERMES, from the two PID controllers to the wall detection sensors.

VII. RISK ASSESSMENT AND MITIGATION

Since the final subproject is the focus of the project as a whole as this semester comes to a close, risk assessment and mitigation for the HERMES maze-solving robot will be given the most attention in this section.

Q-learning is the primary ingredient to the maze solving algorithm. The risk associated with Q-learning would have been high if it not implemented correctly. The likelihood of this not working was very low, because our main focus was to reinforce the reinforcement learning algorithm. The impact that this module will have to on the entire project, if not completed, was very high since the main goal is to prove that artificial intelligence can solve the maze. A mitigation plan would be to use the flood fill algorithm to solve the maze instead.

The efficiency upgrade feature allows the Q-learning algorithm to be more responsive and immune to error during exploration phase. The risk associated was low and the likelihood of this not being done was also low. The impact on the entire project was very low since it was used as an upgrade on the Q-learning algorithm. The migration plan would be to not use this module at all if it fails.

Straight horizontal movement is required for HERMES to not run into the left and right walls. The risk for this to happen was medium to high since the motors can fail at the last moment or the battery power might be low for max efficiency or the encoders can pick of external noise. The impact on the rest of the project for this module not working was medium to high as well, because if the robot runs into the left or right wall, the learning algorithm can be interrupted and possibly fail entirely. The mitigation plan would be to reverse every time it runs into the wall and correct its position/angle.

Clean 90 degree turns are also important in order to explore the maze and get back from a dead end. The risk associated with this module was low to medium. The likelihood of this not working was also low to medium. The impact on the entire system was medium, because if the robot was not making complete 90 degree turns, it can run into a wall due to the error in the angle. The mitigation plan was to have and IMU constantly polling the angle or have a correct function to fix the position/angle.

Wall detection was very important since it tells the algorithm where all the walls are to map the entire maze. The risk was medium to high, since the bot can get lost during the execution phase if it does not keep track of the walls. The impact of this module not being completed on the entire system was medium to high, since the robot will start acting like it was in the exploration phase even during the execution phase. The mitigation plan was to make full stops at every step to allow enough time to register the walls.

Wall avoidance was very important and works in conjunction to the straight horizontal movement module. The risk of this module was medium to high and the likelihood of this not being done was low to medium. The impact on the entire system was high because if the robot cannot avoid colliding with walls than it will fail at exploring and solving the maze. The mitigation plan would be to retrace its steps and correct itself using a custom correct function with the help of the IMU.

Dead end detection for stops was used to make a quick reaction for not running into the wall in front of the robot. The risk was low to medium since it just needs a custom function to determine a wall in front and make a stop before running into the wall. The impact for not completing this module was medium. The mitigation plan would be to give two IR object detection signals, one a little earlier than the other to confirm that a wall was coming up and be ready to stop.

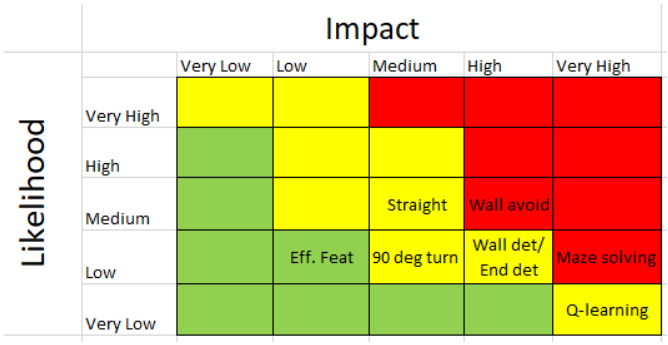


Figure 6. Risk Assessment Chart

VIII. DESIGN DOCUMENTATION: FALL 2017

This section discusses the design philosophies and the stories of the two subprojects developed in fall 2017.

A. FPGA-based ANN

The design of an FPGA-based artificial neural network which would perform optical character recognition was known to be a challenging task from the start of the project, in part because none of the members of this team were comfortably exposed to artificial intelligence prior to working on it, and in part because this line of research is new and gaining interest and lacks much documentation as a result. The challenge, while present, was to be countered by a large number of hours worked per week.

Within the first month of the project, the volume of research references this team had used to in the development of the FPGA model filled two pages. This team combined several academic papers describing floating point arithmetic units in order to create individual floating point addition and multiplication modules.

For the activation function module, papers describing activation function approximations in software were manually referred to in order to develop a Verilog model of the same approximations.

Still more papers were referred to in order to fully understand the mathematics and structure of neural networks and the use of back propagation within them – again using papers focused on software implementations.

Almost all of the work done on this model that was based on research was performed by transcribing ideas present in software research to something useable in Verilog. That said, the combination of those ideas and the creation of novel ideas for the development of this model was the primary work this team performed.

This team finished nearly all work required to make a fully functional Verilog model of a back propagation neural network that would run in simulations.

The completion of the Verilog design was likely by the time this team came upon the realization that the model developed was too complex for upload into the FPGA this team had access to.

Efforts were made to reduce complexity and make the Verilog program work with available resources, but it proved impossible. The FPGA models required to run the Verilog model cost in the range of $3,000 which fell far outside this team's budget. As a result, the Verilog design was left behind.

Nevertheless, the knowledge gained in creating a neural network within a Verilog system was enough to make this team comfortable with the concepts of artificial intelligence in general, and that level of comfort helped with both of the following subprojects.

B. Optical Character Recognition ANN

Optical Character Recognition (OCR) is a problem can be approached from various different directions, but the two most popular are digitization and feature extraction.

In digitization, a letter or number is converted to a matrix of 0's (white pixels) and 1's (black pixels). In doing so, a picture's entire pixel array is iterated through and first converted to grayscale, allowing the conversion of the character 'A' from step (a) to step (b) as seen in *Figure 7*. At this point, the pixel values (in binary form thanks to the grayscale pre-processing) are simply extracted into a matrix which is ready to be used as training/testing data by the OCR system. Another equally popular approach is Feature Extraction.

In the process of Feature Extraction, certain features/qualities/traits are designated, and every image is "taught" to the Artificial Neural Net via an abstraction of these features. These features may include center-of-mass (of the black pixel distribution), segmentation, and conversion to contours and/or vectors. Although the teaching approach is varied, the idea of a set of neurons (784 neurons specifically, assuming a 28x28 input image) learning

to recognize characters based on carefully calibrated/tested weights, remains the same.

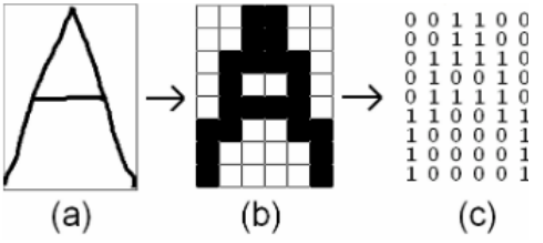


Figure 7. Digitization of the Letter 'A' [7]

Modeling a solution for the OCR (Optical Character Recognition) problem (via Artificial Neural Networks) is seemingly straightforward, but in reality requires a lot of preparation to ensure that both teaching and testing of the system is undertaken with as similar and controlled parameters as possible. An application of controlled parameters does not necessarily take away from an ANN's ability to predict output for real-world test cases.

In the first attempt at developing an OCR system on a Raspberry Pi using a C++ ANN, learning and testing parameters were not controlled, thus leading to discrepancies and high levels of inaccuracy for traditionally written numbers between 0-9.

After extensive debugging and exploring a myriad of different image pre-processing techniques and the order in which they are applied, it was concluded that the issue was not on the validation side, but rather the teaching/training. The stem of this issue arose from the selection of the OCR's training data. Here, the MNIST database was used (a subset of the NIST database), which included 60,000 training images and 10,000 testing images.

As development of the preliminary OCR system continued, it became obvious that the high quality of the MNIST database (a result of scanning all handwritten numbers), could not be satisfactorily reproduced using the Raspberry Pi camera module.

In the next iteration of the OCR prototype, the training data was generated under conditions as similar to the validation scenarios as possible. This involved, taking training/testing pictures of handwritten numbers with the same camera (RPI camera), under similar lighting conditions, and a standardized/controlled focal length.

It was impossible to replicate the entire array of 70,000 images provided by the MNIST database in a timely manner, but equipped with a clear, simple, and capable Neural Network, training was performed with far fewer images (~50 different samples per letter).

Once the custom dataset was developed using volunteer data from various CSUS Engineering students, they were compiled into a text file, formatted using PowerShell (and later Bash) scripting, and were finally inputted to the C++ OCR ANN. The resulting tests were highly accurate and provided accuracy levels above 87% when sent in a randomized set of 100 numbers (these were not used in the training dataset).

IX. DESIGN DOCUMENTATION: SPRING 2018

The HERMES project is a system designed to intelligently solve mazes while providing an outlet for education in robotics and artificial intelligence to students. HERMES, otherwise known as the Hyper Expedient Robotic Maze Extraction System, is comprised of three primary feature. All three are crucial and interlinked, and they can be further broken down into finer features.

In order to develop a learning robot kit, which can be cost-effective and provide itself as a tool to increase Math and Science literacy and interest levels among K-12 students, a robot capable of such features must first be developed.

The IEEE Micromouse competition provides a framework within which can this product can be developed. The micromouse competition assesses the agent's (robot such as HERMES) ability to solve and navigate a 16 x 16 cell maze (where each cell is 18cm x 18cm), as seen in *Figure 3*. Each cell floor space is 16.8cm x 16.8cm, while the walls (5cm height) add an additional 1.2cm (wall thickness), for a total of 18cm x 18cm per cell (5% tolerance assumed) [8].

The HERMES agent, makes use of competition guidelines and outcomes to provide a reasonable and comparable measure of our algorithmic success. The competition allows HERMES, a single run-through of the maze to encode the walls and obstacles in its memory and determine the fastest route from any given corner of the maze to the center. During this first run-through HERMES will "learn" how to avoid obstacles and reach the center of the maze within optimal time. While most agents apply a Flood Fill algorithm, or various other directed maze-solving algorithms, HERMES will instead feature a Reinforcement based learning system called, *Q-Learning*.

Hermes makes use of the Q-learning algorithm to navigate to the end (goal) point. Several auxiliary functions/tools have been provided to HERMES to improve its exploration, solving, and traversing efficiencies. An algorithm such as Q-Learning requires precise sensory inputs and relevant outputs to navigate the maze without collisions/other time wasting events. The movements will be provided to the motor controller in the form of instructions to move cell by cell (18cm x 18cm).

A. System Overview & Implementation

The HERMES system's main features can be split into the three following categories: Mind, Body, and Senses. Each of these pivotal categories brings with it its own subset of features and components which work harmoniously with the rest of the HERMES system.

B. Mind: Q-Learning

The HERMES agent must be able to solve the maze given a 16x16 matrix of 4-bit values (cardinal wall locations in every cell). Because this team's goal is to further explore and apply concepts of artificial intelligence and learning, the HERMES agent has instead been equipped with a reinforcement based, Q-Learning algorithm.

The Q-Learning algorithm served as the framework upon which the HERMES Mind algorithm was developed. Q-Learning is an adaptation of Markov Decision Processes, which is a decision algorithm that, in short, looks at all possible actions and chooses the one that will lead to a goal in the shortest path. The Markov Decision Process alone is impossible to implement without providing all possible states and action values within those states. For a sufficiently large or complex system, defining those values ahead of time becomes unruly.

The Q-Learning algorithm provides a means by which the shortest path can be determined with trial-and-error experimentation. The agent can arrive at the end, then determine which states can lead directly to the end, then determine which states lead to goal-adjacent states, and so on until it finds a path that leads from the beginning to the end. When it traverses the action matrix, it will always take the action with the greatest reward value.

To do this in practice, discrete states must be quantized to simplify a real-world system that consists of effectively infinite states, then define reward values for actions within those states. In practice, exploring and finding the end of a maze through trial and error is an arduous process, so further alterations were made to shorten that process.

The first of those changes was the implementation of a predefined reward matrix. Within the context of the IEEE Micromouse competition, which is a predefined set of rules for maze generation and solving, the end goal is in a known location. Furthermore, the start position is also known – it is in one of the four corners. Thus, the reward matrix makes use of these restrictions and places the goal squarely in the center of the maze.

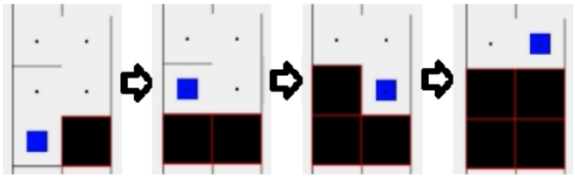


Figure 8. Dead End Avoidance

Another attribute added to the Mind is dead-end avoidance. This is described most clearly by *Figure 8*. When a dead end cell is visited – that is to say when three walls are detected adjacent to the agent – HERMES internally walls off the cell. This is demonstrated visually in *Figure 8* with the black

square overlaid onto the maze cell. Since these dead-end cells are virtually counted as walls, real or virtual, dead end corridors are also blocked off. HERMES will never revisit a cell that has been marked as a dead end.

A third alteration to the algorithm changes the reward value of every space which is visited by HERMES through a process called reward decay. This is a simple alteration which takes the current reward value of a space that HERMES just left and decrements it by one (1). This encourages wider exploration by making locations that have already been visited less appealing to HERMES and placing choices toward new locations at relatively higher values.

The final alteration is one called assured decisiveness. This became an important alteration to counteract an emergent property of reward decay that we termed *indecisiveness*. In long corridors that needed to be revisited multiple times, since the reward matrix was altered with every pass of HERMES, occasionally HERMES would "change its mind" about the direction it was going and double back to where it had just been. This became a very inefficient property, and it was counteracted by artificially placing spaces that were just left by HERMES at lower reward values than given by the reward matrix.

In all, this set of changes formed the HERMES Mind, which now efficiently solves mazes upward of 128x128 in simulations.

C. Body: Chassis & Motors

The second extremely important factor of the HERMES system is the chassis. This chassis holds all components of the Mind (processor) and SENSES (sensors and motor controller).

Throughout the thirteen (13) different iterations of HERMES, the chassis design has changed considerably. While the first couple versions were simply built to test features and made use of lightweight, yet bulky balsawood. More recent versions of the chassis have made use of 3D printed custom component casings, full 3D printed chassis, and plexiglass.

On top of holding all necessary components, the chassis must also be small and compact to help ensure crisp right/left turns, avoid inadvertent collisions (wastes time), and accelerate/move through specialized maze portions such as 'stairs'. The most recent version (v13) of the HERMES chassis features 2 plexiglass sheets with 1.25 inch separators between them to allow sufficient space for components.

Another major design idea with the HERMES chassis is the wiring, which should be clean and tucked away to prevent snagging on the maze itself or the agent's motors/wheels. Often times the wiring from the battery is the bulkiest because of custom solder points to split the power which make the entire wiring scheme a mess.

The second pivotal factor of the Body, is the motor system. Again, throughout the various versions of HERMES, many different motors were used, DC (v3-v8, v11, v12), Steppers (v1, v2, v13), and even continuous servos (v9, v10). While DC motors didn't have necessary torque without a proper gearbox, most the stepper motors were too slow. In addition, the DC motors required one H-Bridge (bulky add-on components to provide differential speed control for these motors), and the stepper required two H-Bridges. The continuous servo motors that were once used didn't require any H-Bridges, but synchronizing two servo motors was extremely difficult.

It is important to have fast motors to keep run/completion time competitive, but this factor must be balanced with the torque necessary to move all on chassis weight. This was a daunting task given that the agent's weight was constantly changing during initial builds, and the DC motors used for many builds, required constant gearbox tinkering to achieve the necessary torque to move the constantly changing chassis + component weight.

D. Sense: Spatial Awareness and Controls

The HERMES agent uses a variety of sensors to enable spatial awareness. The two most important goals of the sensors are to first, help the agent safely navigate the maze. And Second, to

provide the processor with a 16 x 16 4-bit matrix of the cardinal wall locations in every cell of the 16 x 16 maze. Despite the level of innovation and redesigning required of the other portions of the HEREMS agent, the senses were relatively hashed out early in the project and have stayed constant, except for the addition of an IMU.

Initially proximity based, Ultrasonic Sensors were used because they provide a feedback of how far away the walls are, a priceless piece of information. However, they are difficult to multi-process with the motors, as well as highly susceptible to external environment noise. This random environmental noise can interrupt the relay of sonar pulses being emitter by the sensor and being looked for by the receiver. As such, the sonar sensors were quickly discarded in favor of Infrared Sensors.

The IR sensors are great for the task of detecting walls because they operate at a specific distance (distance between agent and wall) based on potentiometer setting. This allows for them to be easily fine-tuned as this distance between the agent and walls change because the agent's dimensions are still changing throughout the various chassis iterations. IR sensors are also relatively low weight and not as bulky as the proximity Ultrasonic sensors were.

Initially, encoders with a resolution of 3.6 degrees were used in tandem with 90mm (diameter) wheels. This provided an encoder tick at every 2.83mm which helped keep track of wheel displacement. However, encoders are a tricky add-on, especially when the chassis and motors were constantly changing during the earlier phases of the project.

As such, an alternative was found with the micro-steppers being used now. The new micro-steppers (v13) use a built in step count to control their movement. This allows a constant level of precision because each step (degree of rotation) of the motor is recorded and accounted for. The current steppers being are based on a 4096 steps per complete revolution system.

Despite their complications, the second viable version of HERMES (v12) uses magnetic encoders to keep track of the motor revolutions. The magnetic encoders on this version of HERMES provide about 680 ticks per revolution. This level of precision is important for the Pulse Width Modulation (PWM) and Proportional Integral Derivative (PID) functions to control HERMES' speed. Despite the level of precision provided by both of these motors, there is still unwanted movement as the agent travels through the maze.

This problem is somewhat mitigated by application of the BN005 IMU. An IMU provides the agent with useful information such as gyroscope, angular, and rotational acceleration. The IMU helps the agents move in straight lines without deviating due to discrepancies in the continuous servo motor tuning. By detecting for a sudden angular acceleration while performing a specified movement, the motors can use the information provided by the IMU to correct motor rotation speed and return to a best course for optimal completion time.

Controlling the motors has generally been straightforward while using Python. Controlling and correcting their movement using encoders and sensor data is a different story altogether. There have been various techniques used such as wall following, "bounce-back" correction, and more, but the most effective seem to be to use an advanced variation of "bounce-back" correction (for the v12 DC HERMES) and simple step correction (for v13 Stepper HERMES).

The version 12 of HERMES uses DC motors as well as sensors which help provide data with which to ensure straight line movement. The sensors provide data when the agent is drifting to one particular side, thus setting off the IR sensor set to a specified distance. These sensors coupled with the magnetic encoders allow HERMES to determine when it is drifting towards a wall and then employ the encoder data to perform a PID calculation to correct its motor speeds and return to a straight line path. The PID calculation helps the MIND send out appropriate PWM signals to correct the motor path. For example, if the agent is drifting left, the left

motor would be sped up, while the right motor is slowed down to provide the necessary path correction. The IMU is also very effective in helping HERMES determine when it is drifting before the IR sensors are triggered.

The stepper version of HERMES (v13) employs a similar method, but instead of using PWM signals the agent simply reverse specific numbers of steps based on the amount of correction needed by either the left or right motor.

X. DEPLOYABLE PROTOTYPE STATUS

For the current subproject and the final work for the year, this team will discuss the deployable prototype status of the HERMES model in the context of the 2018 design contract. To simplify the discussion, this section will talk about the three main features and then their integration with each other.

A. The Mind

This feature was given three primary requirements by the 2018 design contract: Q-learning implementation, maze solving ability, and the design of several Q-learning efficiency upgrades.

The Q-learning implementation was successful very early on in the semester. The design used the restrictions of the Micromouse competition to develop a reward matrix wherein the elements of the desired goal at the center of the matrix were placed at the highest values and all concentric 'rings' of the matrix extending to the borders were given a decreasing reward value gradient – shown in *Figure 9*.

-8	-8	-8	-8	-8	-8	-8	-8	-8	-8
-8	-7	-7	-7	-7	-7	-7	-7	-7	-8
-8	-7	-6	-6	-6	-6	-6	-6	-7	-8
-8	-7	-6	-5	-5	-5	-5	-6	-7	-8
-8	-7	-6	-5	0	0	-5	-6	-7	-8
-8	-7	-6	-5	0	0	-5	-6	-7	-8
-8	-7	-6	-5	-5	-5	-5	-6	-7	-8
-8	-7	-6	-6	-6	-6	-6	-6	-7	-8
-8	-7	-7	-7	-7	-7	-7	-7	-7	-8
-8	-8	-8	-8	-8	-8	-8	-8	-8	-8

Figure 9. Reward Matrix

The Q-learning algorithm used the reward matrix heuristic to determine which direction it should travel in to reach its goal. Q-learning was thus implemented.

A simulation was designed to allow a simulated HERMES to travel in mazes far larger than the real world maze this team used for testing the robotic model. The simulated model always reached the goal, as expected, meaning that the maze solving feature was fully developed.

Efficiency upgrades were necessary in order to account for some of the wasted time spent revisiting dead ends and corridors, and time spend going back and forth within corridors. These features were fully implemented, along with additional upgrades to make finding the goal a shorter process as well.

B. The Body

This feature had two primary requirements, which are key to being able to navigate an actual maze: horizontal movement and clean ninety degree turns.

The design of the chassis is important for these two things – for example a larger pair of wheels makes more precise motion easier to attain, and wheels placed approximately in the center of the robot allow ninety degree turns to occur in the center of maze cell. These considerations were kept in mind with later physical models.

This body feature, while physical, occurs primarily in software through PID controllers. Straight motion was achieved through the use of encoders. Every tick of the encoder is recorded through hardware interrupt functions, and the PID controller adjusts motor speeds to account for differences in the speed of the wheels. This function makes use of a master and slave model, where one motor is given a fairly constant speed and the other must adjust to meet that speed. The robot moves very straight now, thus the straight movement requirement is met.

Ninety degree turns occur within a PID controller as well – this one using an IMU to determine absolute orientation and encoders to make the wheels turn at the same speed. Ninety degree turns have been achieved, and not only that but ninety degree turns relative to starting position have been

achieved too. This means any error accumulated upon stopping the robot is accounted for. This requirement was met.

C. The Sense

This feature allows the HERMES Mind to detect obstacles and walls in the real world. The requirements for this particular feature were in wall detection, wall avoidance, and end detection.

All of these requirements have been met by using digital IR sensors. Four such sensors are calibrated to detect walls on all four sides, detecting up to half a cell away to ensure wall detection and avoid detecting walls in another maze cell.

Three sensors are calibrated for very close range wall detection. Two of them detect walls on the front left and right ends to tell HERMES that it is about to hit a wall on each side, while the one in the front tells HERMES that it is about to run head on into a wall.

These sensors work together to accomplish all sense features required by the 2018 design contract.

XI. PROTOTYPE MARKETABILITY

The HERMES deployable prototype would need a user interface to be a marketable device. The amount of work required to achieve this module is three to four months of integration. The hardware will need an upgraded chassis material, similar to ABS plastic. Sensors, such as IR and IMU, will need to be embedded inside the PCB design. And most importantly, the user interface needs to be accomplished by adding a touch screen or keypad onto the system to allow students to interact with HERMES. To accommodate the newly added hardware, the software must also be updated, especially for the user interface module. The software will include prompting the user to start the program and run it for a certain time before it stops. But the interface would also give feedback to the user for how HERMES makes decisions when roaming around the maze.

XII. CONCLUSION

The spring semester of Senior Design has been spent developing a tool to help mitigate the falling Math and Science literacy levels in America. In order to achieve this, a product with AI and Learning based robotics had to first be developed.

Using the IEEE Micromouse competition as a framework within which this team could develop a product (HERMES) has been challenging but has proved to be perfect for the desired end product. The semester long exploration into robotics and applying AI to them appropriately has provided useful information about how to develop a cost-effective tool to increase k-12 Math and Science literacy levels.

In developing HERMES to solve the Micromouse maze, many discoveries have been made about motors (Steppers, DC, Servos), sensors (IR, Sonar, Proximity, IMU), and Control systems (PID, PWM). This information has been effective in helping this team developing an agent to solve the Micromouse competition, and has also provided useful in determining the best parts (cost-effective vs. performance).

The thirteen different builds of HERMES, some of which are show in Appendix H, have shown this team that developing a physical agent capable of using the sensors available to properly navigate and record the maze is a process that is more or less the same every time. Starting with high quality components is helpful, but difficult with a budget that is never large at any given time.

The Python based Mind will work with any hardware system as long as the control functions for eighteen centimeter forward movement and ninety degree turns exist. Integration of the maze solving algorithm into a hardware system is as simple as making those two control functions for movement. Despite completion of the Mind (maze solving algorithm based on Q-Learning) early on in the spring semester, finding reliable motors that were capable of performing precise movements for the required movements made this project a semester-long endeavor.

After finally developing two different agents with precise and accurate controls, it was relatively easy to integrate the Mind (Q-Learning algorithm). The agent is using Q-Learning principles and the related reward matrix to influence its decisions (to turn or continue straight) whenever possible.

XIII. REFERENCES

[1] Gregory Camilli, Explaining the National Assessment of Educational Progress 2013-2015 Mathematics Decline. https://nsf.gov/awardsearch/showAward?AWD_ID=1641257&Historica lAwards=false

[2] Drew Desilver, U.S. students' academic achievement still lags that of their peers in many other countries. http://www.pewresearch.org/facttank/2017/02/15/u-s-students-internationally-math-science/

[3] U.S. Department of Education, Institute of Education Sciences, National Center for Education Statistics, National Assessment of Educational Progress (NAEP), 2009 and 2015 Science Assessments. https://www.nationsreportcard.gov/science_2015/#gaps/chart_loc_1?gra de=12

[4] Sutton, Richard S. and Barto, Andrew G., Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998.

[5] Delony, David, Micromouse Robot Solves Maze in an Instant. Walyou. http://walyou.com/robot-mouse-maze/

[6] Pozos-Estrada, Adrián, Gómez, Roberto, and Hong H.P. Use of Neural network to predict the peak ground accelerations and pseudo spectral accelerations for Mexican Inslab and Interplate Earthquakes. March, 2014. http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0016-71692014000100004

[7] Araokar, Shashank. Visual Character Recognition using Artificial Neural Networks. https://arxiv.org/ftp/cs/papers/0505/0505016.pdf

[8] Sites.ieee.org, 2018. [Online]. http://sites.ieee.org/r1/files/2013/03/2013-Region1-Micromouse-Competition-Rules.pdf

[9] http://automouse.sdsu.edu/images/block%20diagram.jpg

[10] http://eeshop.unl.edu/pdf/Stepper+Driver.pdf

XIV. GLOSSARY

Artificial Neural Network (ANN): A model for learning algorithms that mimics the design of biological brains.

Optical Character Recognition (OCR): A system which inputs an image of text and determines which characters are present.

Field Programmable Gate Array (FPGA): A piece of hardware used for testing Verilog hardware description language modules.

Artificial Intelligence (AI): A general term for any system that adapts to inputs to generate better outputs. Better in this case is defined by the designer.

Activation Function: A function used to collapse values being input into a system to a number between 0 and 1 or any other range of values desired by the system.

Micromouse: An IEEE competition standard which defines a maze through which robots explore and discover the end as confined by IEEE rules.

XV. APPENDIX A. USER MANUAL

There is not a specifically designed User Interface for our HERMES agent at this time because the goal of this semester was to develop the AI/Learning based robot which would then be equipped with additional tools and features such as a User Interface to assist K-12 students and educators.

However, despite this, there are clear modular functions which allow the HERMES agent to be controlled to high level of precision. Thus a prospective user can specify the exact amount of distance they would like to see HERMES travel in any specified direction, or perform certain movements such as turning in place. These can be done on both version 12 (DC motor based) by setting distances and on version 13 by setting the desired number of steps to go in any direction.

These functions also allow the user to also specify the learning parameters used by the Mind, if they would like to change any specific aspect of the Q-Learning efficiency and/or fiddle with the level of indecisiveness.

XVI. APPENDIX B. HARDWARE BLOCK DIAGRAM

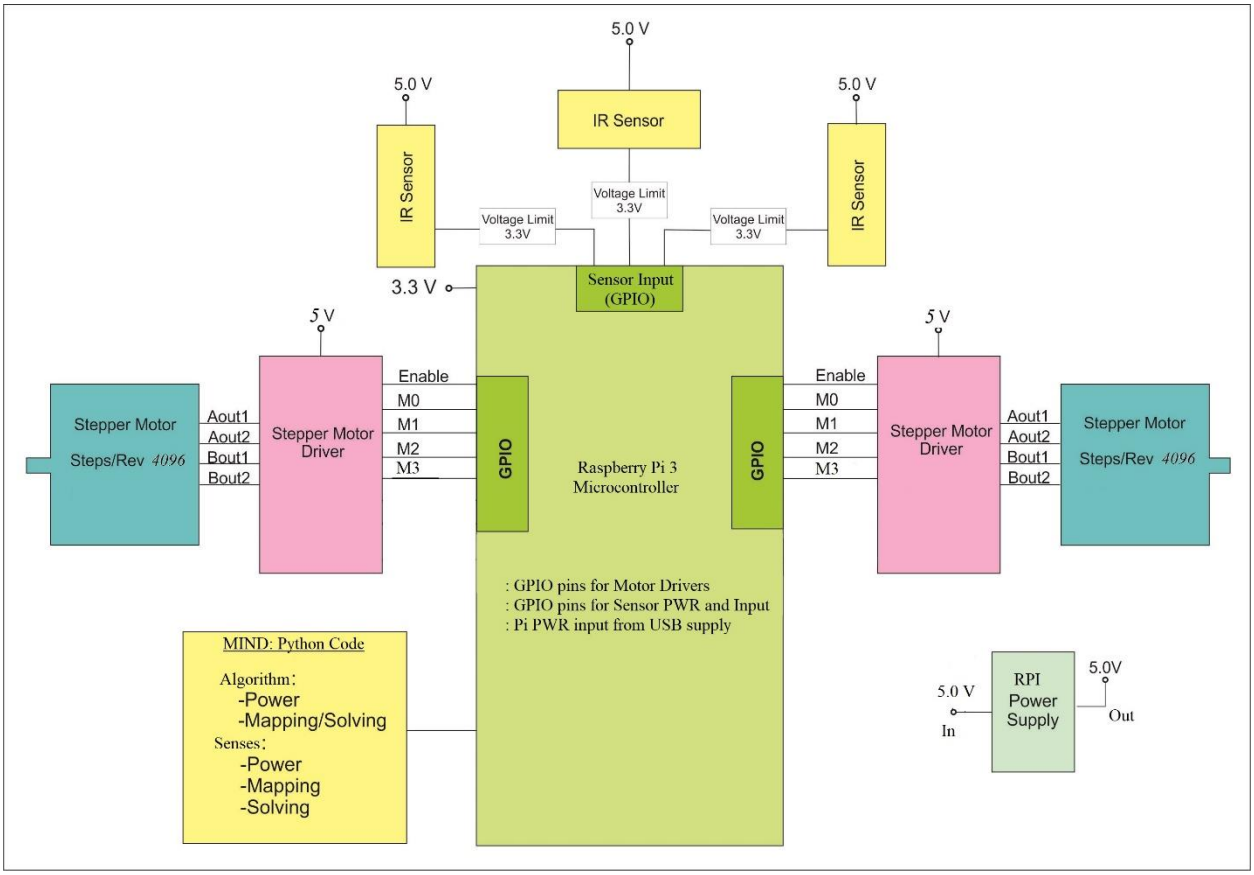


Figure 10. Hardware Block Diagram (Adapted from [9])

XVII. APPENDIX C. SOFTWARE BLOCK DIAGRAM

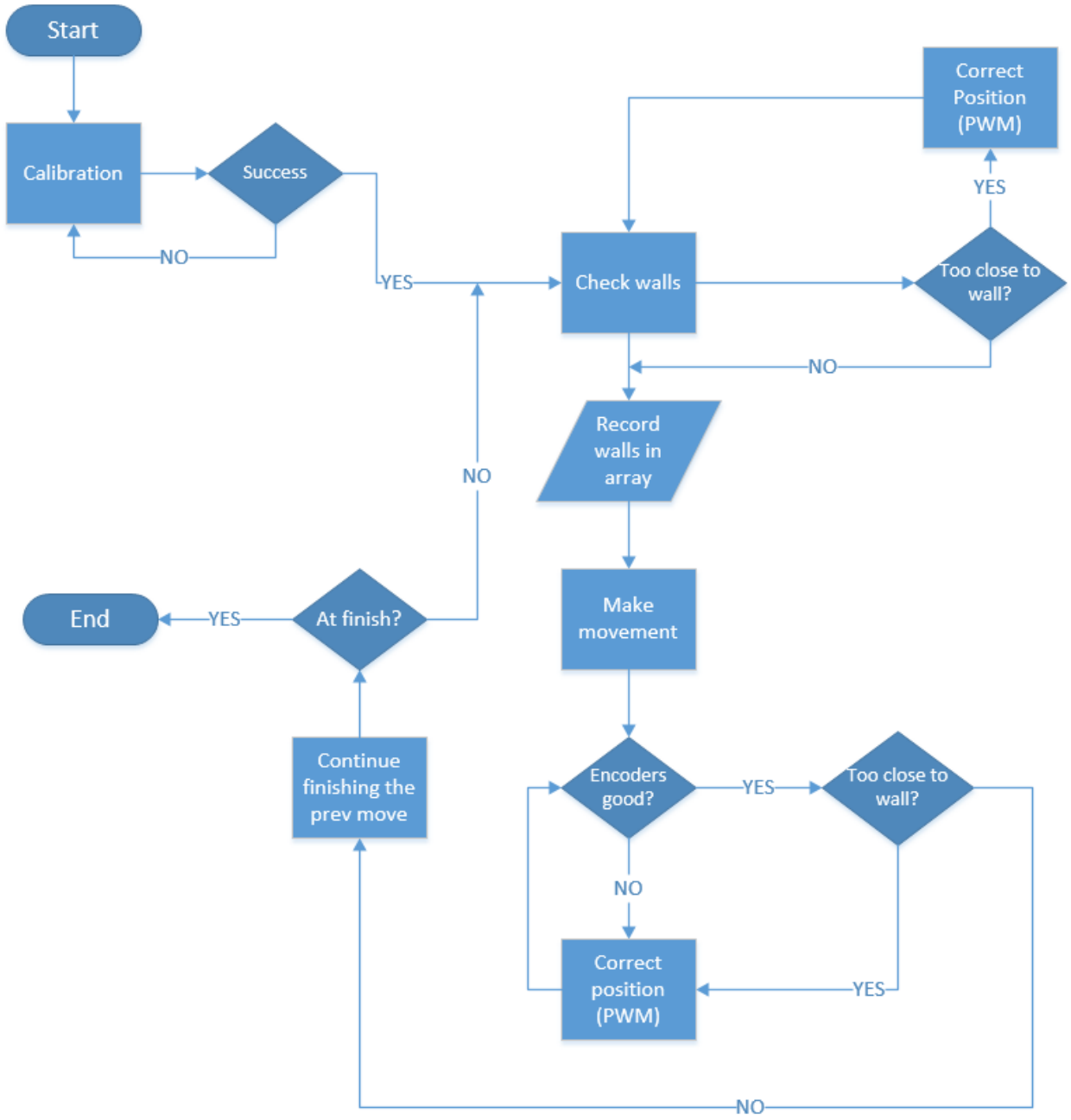


Figure 11. Software Block Diagram

XVIII. APPENDIX D. MECHANICAL DIAGRAMS

There were no relevant drawings, load calculations, or other documentation for the chassis build as it was done as an iterative process based on the varying parts used throughout the semester. However, the DC motors w/ encoders (version 12) and micro-steppers (version 13) do have some relevant documentation as seen below.

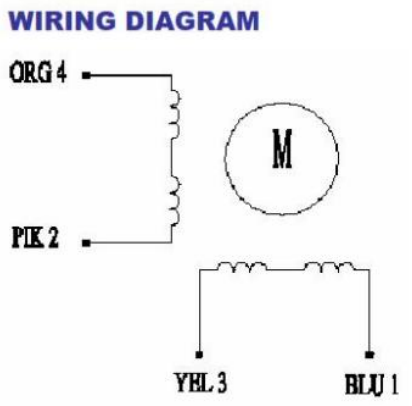


Figure 12. Wiring Diagram for Stepper [10]

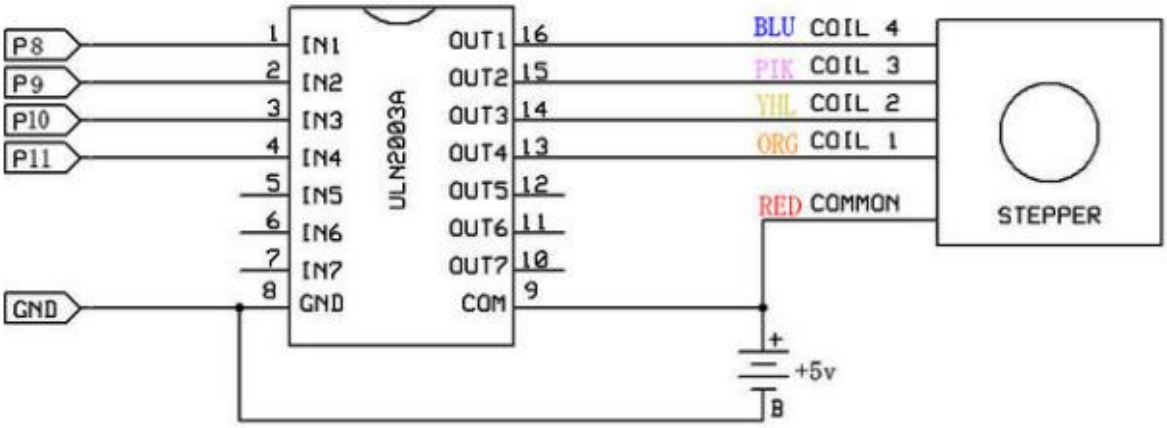


Figure 13. H-Bridge Wiring Diagram [10]

XIX. APPENDIX E. VENDOR CONTRACTS

This team received no help from vendors or off-campus agents in completing this project. However, there are organizations in the greater Sacramento area that are involved with robotics, education, and IEEE. These organizations, such as Schilling Robotics-FMC Technologies, could have proven to be useful resources.

XX. APPENDIX F. RESUMES

BREEANA D. PROFFIT

Proffit@csus.edu

SKILLS

- Logic Design
- Verilog and VHDL
- Java/C/C++/x86 Assembly
- Adept at Debugging
- Leadership
- Critical Thinking and Problem Solving
- Time Management
- Mathematics

EDUCATION

January 2014 - May 2018

Bachelor of Science: Computer Engineering

California State University, Sacramento, 6000 J Street, Sacramento, CA

- Grade Point Average: 3.5
- Received Dean's Honor Roll commendation seven times
- Member of Tau Beta Pi honor society - placement in the top 12.5% of students in my major
- Coursework includes Advanced Logic Design, Advanced Computer Organization, Microcomputers & Interfacing, Signals & Systems, Networks & Internets, and Data Structures & Algorithms.

2013

Associate of Science: Natural Sciences

Sierra College, 5000 Rocklin Road, Rocklin, CA

EXPERIENCE

- Four months as AVID tutor at George A. Buljan Middle School in 2013 | 100 Hallissy Dr, Roseville, CA 95678 | Coordinator: Luanne Harold – lharold@rcsdk8.org
- Three years as freelance college level mathematics tutor from 2015 through 2017. Worked with students to help them understand algebra, trigonometry, calculus, and differential equations.
- Four months as a teacher at Huntington Learning Center, 2017-Present | 2216 Fair Oaks Blvd, Sacramento, CA 95825 | Manager: Liz McCoy - sacramentoCCCA@hlcmail.com

Sudhakar Alla

Sacramento, California, 95835
916-765-0335
sudhakaralla1@gmail.com

WORK EXPERIENCE

MENTAL SCHOLAR
Keynote Architect, Jun 2012 – Present

- Developed Keynote Presentations to be integrated into an iPhone app that aims to effectively teach kids Algebra
- Created custom 2D Animations, using Adobe Flash and Apple Keynote, which were embedded into both the keynote presentations and the app's User Interface (UI)
- Developed LaTeX and Perl scripts to render high-resolution text and math formatting for use in the presentations and problem sets used in the app
- Transcribed audio overlay for the presentations and setup Closed Captioning (CC) using Adobe Premier and Google Audio Services
- Created and formatted custom Algebra problems using LaTeX, used in the app and will also be published separately

EDUCATION

California State University at Sacramento, Sacramento, CA
Computer Engineering Candidate, Expected graduation, May 2018

- Transferred from Cal Poly, San Luis Obispo in 2013 to work with Mental Scholar and decided to transfer to Sacramento State
- Coursework includes Embedded Systems, Computing Theory and Programming Languages, Webpage Development, Software Engineering Principles, Logic Design, Linux/Unix, and Signals Processing
- Active member of ACM (Association of Computing Machinery)
- IM Basketball Captain (February 2016 - Current)

American River College, Sacramento, CA
3D Technical Director Certificate Candidate, Expected graduation, Jul 2018

- Finishing the following Art New Media (ARTNM) certificates: 3D Technical Director, 3D Animation, and 3D Rigging Technical Director

PROJECTS

- Motion Detection/Facial Recognition with Raspberry Pi 2 : Implemented using OpenCV graphics libraries • Loaded multiple test sets to help identify specific individuals • Combined with a remote controlled Dolly as a security device • Python scripts used to send matched facial shots to a server
- Short production of my own 3D animated film : 3D modeling/texturing/rigging • Custom FX effects and final production lighting and rendering

ADDITIONAL SKILLS

- Java/C/C++
- Verilog/VHDL/Prolog
- Adobe Suite proficient (Premier/Flash/DC)
- Perl/MEL/Python proficient
- Experienced working with LaTeX scripting (Math Formatting)
- Autodesk Maya and Pixologic ZBrush (3D Modeling/Rendering Software)

VIKRAM SAROAY

vikramsaroay@gmail.com

Objective

To obtain an internship position, as a Computer Engineer, where I can apply and improve my skills.

Education

COMPUTER ENGINEERING | GPA: 3.6 GRAD. MAY 2018 | CSU SACRAMENTO

- Dean's Honor List (7 semesters)
- Association of Computer Machinery Member
- IEEE Member

RELATED COURSEWORK

- *Programming Methodology* - Developed a java GUI interface that allows users to create and store various transactions, i.e. deposits and withdraws. Used serial I/O to backup and then load user info.
- *Data Structures & Algorithm Analysis* - Tested various sorting algorithms, applied the recursive programming technique to solve common issues such as program execution time.
- *Advanced Logic Design* - Implemented and tested hardware components such as, multiplexors, SRAM, and ALU in various projects and assignments using Verilog and VHDL.

Skills & Abilities

COMPUTER LANGUAGES

- C, C++, Java, Visual Basic, Python, Verilog, VHDL, Unix/Linux
- HTML5, CSS3, JavaScript (Node, React, angular), PHP

COMMUNICATION

- Delivered individual and group presentations and demonstrations on various research topics including, automatic parallelism in software, cyber security, .

Project / Personal Experience

16 BIT SINGLE ISSUE PIPELINED CPU DESIGN / IMPLEMENTATION

- Designed a pipelined data-path and control unit for the 16 bit system. Implemented the design using Verilog. Tested each component and stage individually, and tested the fully working CPU using assembly instructions and monitor.

JAVASCRIPT TABLE HOCKEY GAME

- Used HTML5 in conjunction with JavaScript to a simple a game that lets the user control a paddle with their mouse (event listener) and play table hockey with an A.I. player.

MAX32 (ARDUINO BASED BOARD) CONTROLLED MINI-COMPUTER

- My task in this group project was to wire the circuitry and write one fourth of the code that was responsible for executing the calculator application (displayed on LCD panel and used keypad for user input).

Work Experience

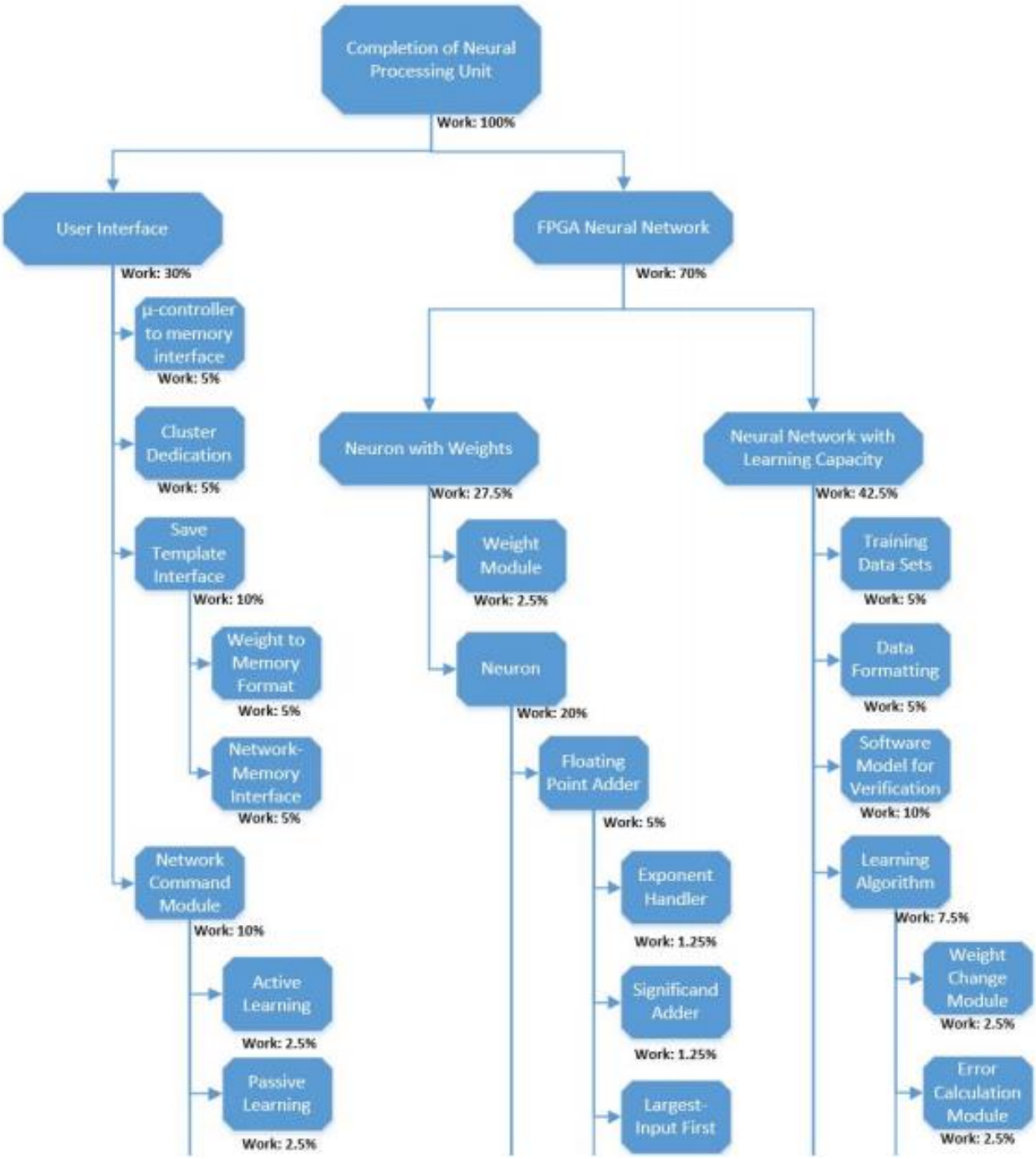
QUALITY ASSURANCE INTERN | DERMVEDA | CURRENT

- Full stack web development using AngularJS/ReactJS, NodeJS, MongoDB, MySQL. Building QA testing software to help with quality of Dermveda apps. Code review of prospective production code. Assistance with dev tools. And used U/UXI.

SALES CONSULTANT | BEST BUY CO. | JULY 2016 – DECEMBER 2016

- Benefited the company by achieving higher than the set hourly revenue of $800 goal per employee, which includes the selling of the main device, necessary accessories needed for the full experience, and best buy services such as Geek Squad.

XXI. APPENDIX G. WORK BREAKDOWN STRUCTURES



XXII.

Figure 14. Work Breakdown Structure: FPGA Model (Part 1)

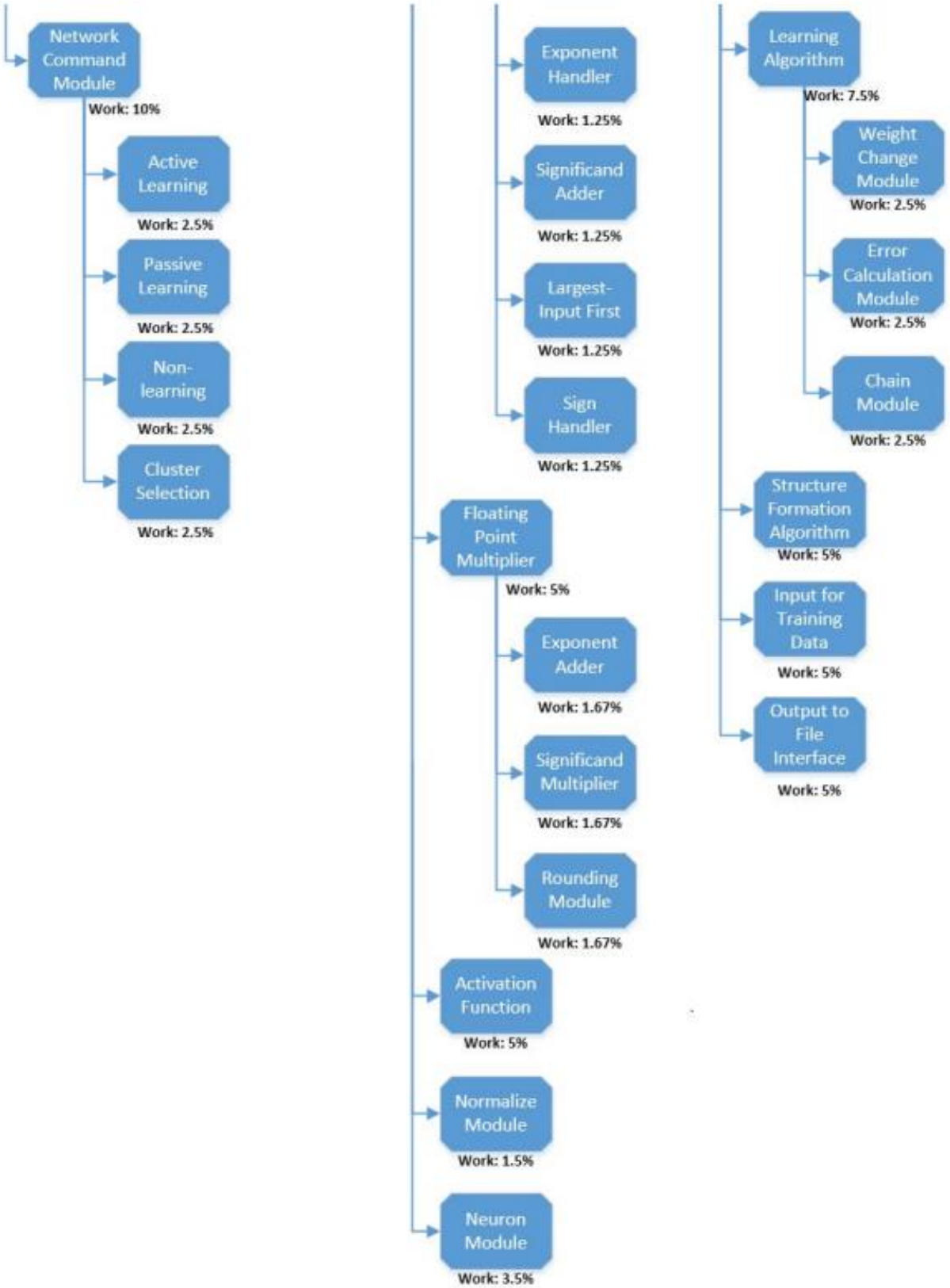


Figure 15. Work Breakdown Structure: FPGA Model (Part 2)

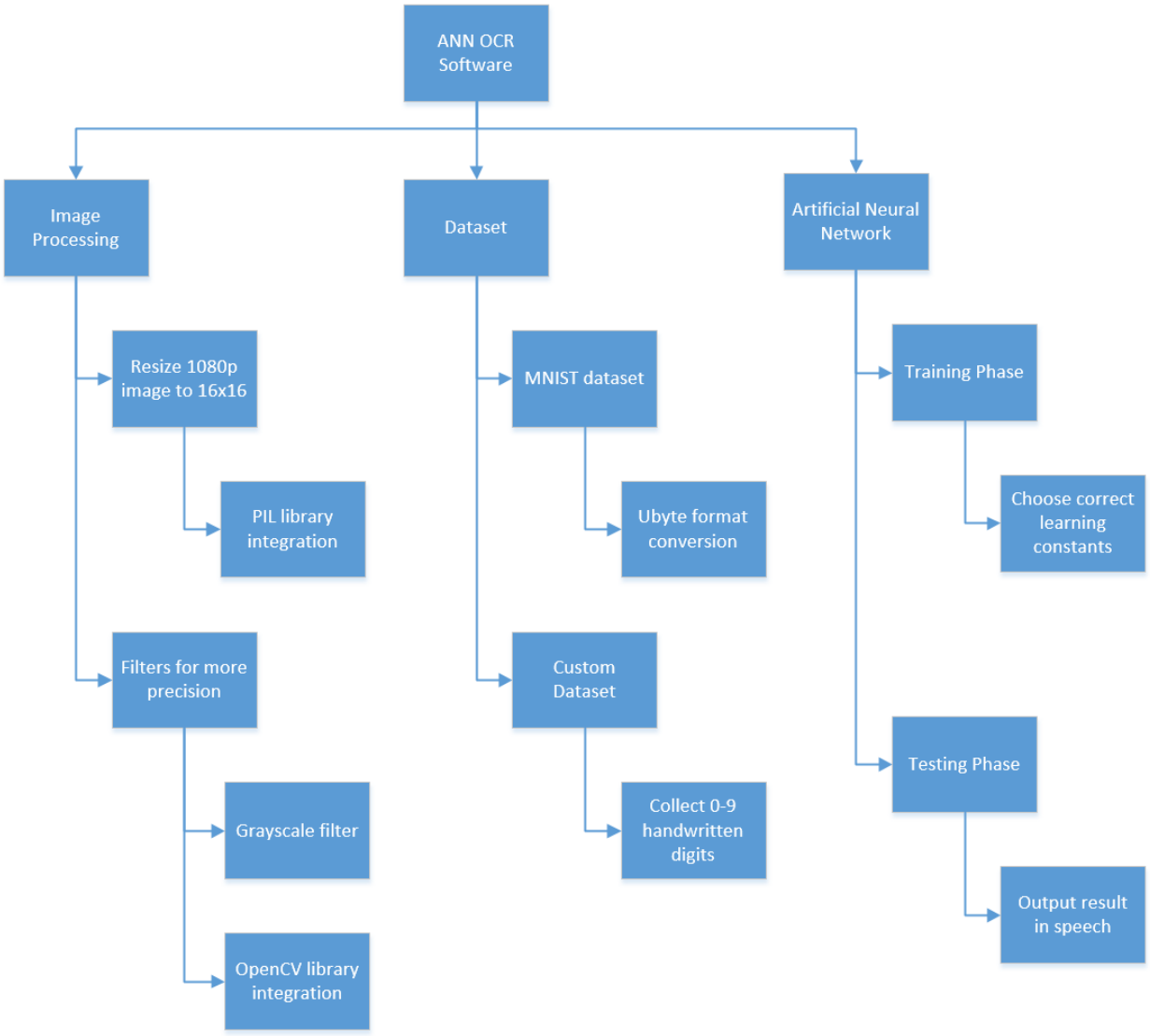


Figure 16. Work Breakdown Structure: OCR Model

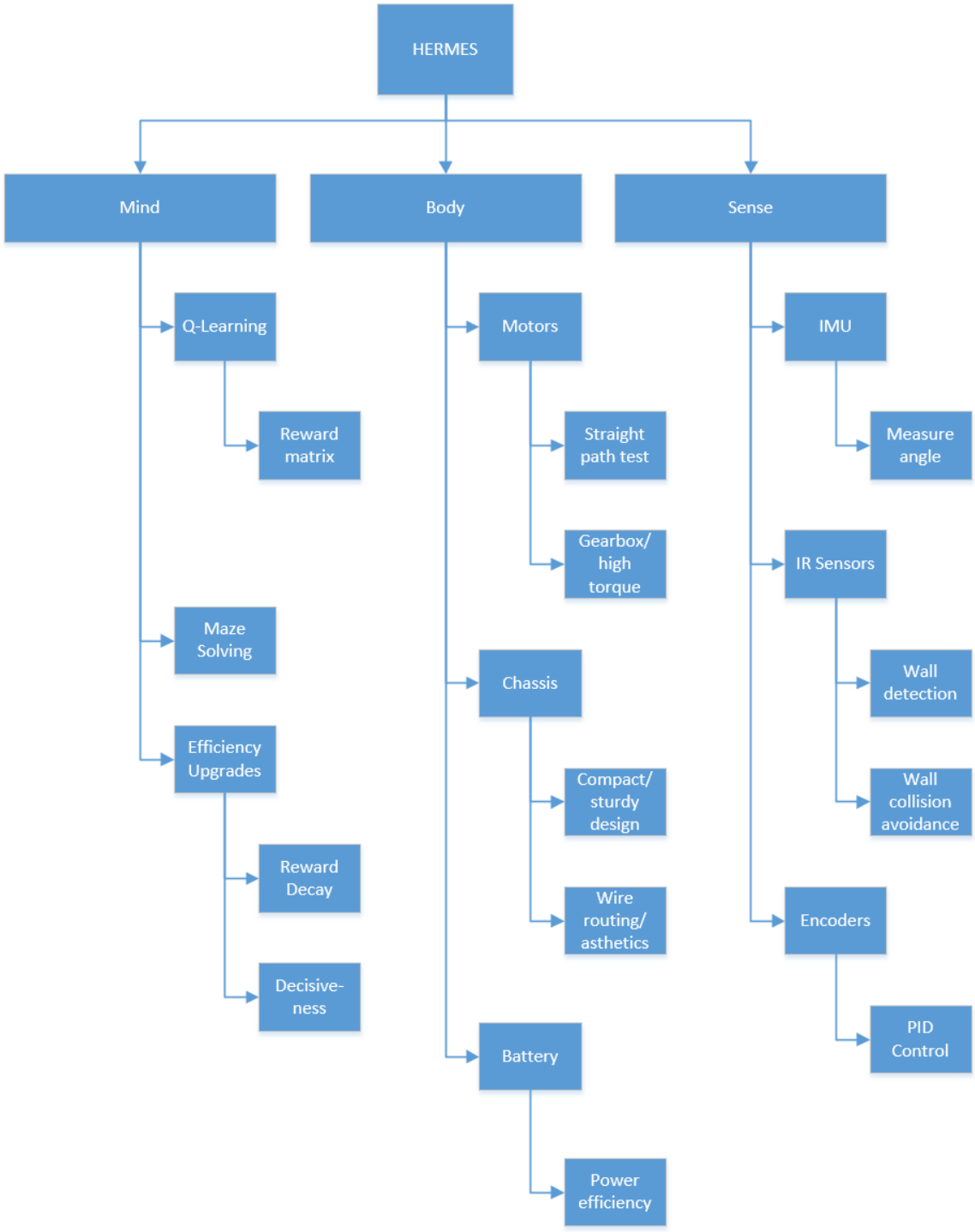


Figure 17. Work Breakdown Structure: HERMES Model

XXIII. APPENDIX H: HERMES MODELS AND IMAGES